

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**Applicant:** Tetsuo Shibuya**Docket:** 14043(JP919990270US1)**Serial No.:** Unassigned**Dated:** December 14, 2000**Filed:** Herewith

For: A METHOD FOR CHANGING A TARGET
ARRAY, A METHOD FOR ANALYZING A
STRUCTURE, AND AN APPARATUS, A
STORAGE MEDIUM AND A TRANSMISSION
MEDIUM THEREFOR

Assistant Commissioner for Patents
Washington, DC 20231

CLAIM OF PRIORITY

Sir:

Applicant(s) in the above-identified application hereby claim the right of priority in connection with Title 35 U.S.C. §119 and in support thereof, herewith submit(s) a certified copy of Japanese Patent Application No. 11 368420 filed on December 24, 1999.

Respectfully submitted,

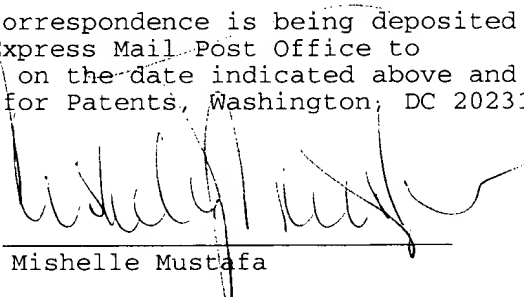

Richard L. Catania
Registration No. 32,608

SCULLY, SCOTT, MURPHY & PRESSER
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343
RLC:ml

CERTIFICATE OF MAILING BY "EXPRESS MAIL"

"Express Mail" mailing label number: EL748591425US
Date of Deposit: December 14, 2000

I hereby certify that this correspondence is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 C.F.R. §1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, DC 20231.

Dated: December 14, 2000
Mishelle Mustafa

CERTIFICATE OF MAILING BY "EXPRESS MAIL" (37 CFR 1.10)

Applicant(s): Tetsuo Shibuya

Docket No.

14043(JP919990270US1)

Serial No.
UnassignedFiling Date
Herewith

Examiner

Group Art Unit

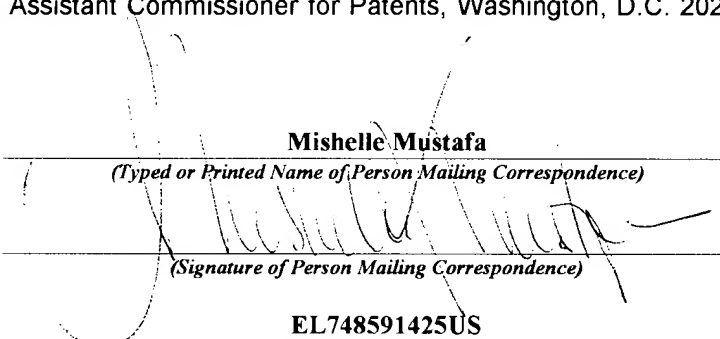
Invention: **A METHOD FOR CHANGING A TARGET ARRAY, A METHOD FOR ANALYZING A STRUCTURE, AND AN APPARATUS, A STORAGE MEDIUM AND A TRANSMISSION THEREFOR**

I hereby certify that the following correspondence:

Utility Patent Application Transmittal

(Identify type of correspondence)

is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 in an envelope addressed to: The Assistant Commissioner for Patents, Washington, D.C. 20231

December 14, 2000*(Date)*Mishelle Mustafa*(Typed or Printed Name of Person Mailing Correspondence)*
*(Signature of Person Mailing Correspondence)*EL748591425US*("Express Mail" Mailing Label Number)*

Note: Each paper must have its own certificate of mailing.

JCT14 U.S. PTO
09/13/00
12/14/00

日 本 国 特 許 庁
PATENT OFFICE
JAPANESE GOVERNMENT



別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日

Date of Application:

1 9 9 9 年 1 2 月 2 4 日

出 願 番 号

Application Number:

平成 1 1 年 特 許 願 第 3 6 8 4 2 0 号

出 願 人

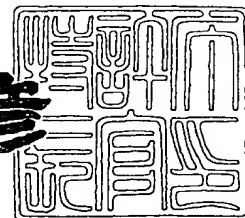
Applicant (s):

インターナショナル・ビジネス・マシーンズ・コーポレーション

2 0 0 0 年 6 月 2 9 日

特 許 庁 長 官
Commissioner,
Patent Office

近 藤 隆 彦



出 証 番 号 出 証 特 2 0 0 0 - 3 0 5 1 3 0 5

【書類名】 特許願

【整理番号】 JA999270

【提出日】 平成11年12月24日

【あて先】 特許庁長官殿

【国際特許分類】 G06F 19/00

【発明者】

 【住所又は居所】 神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ピー・エム株式会社 東京基礎研究所内

 【氏名】 渋谷 哲朗

【特許出願人】

 【識別番号】 390009531

 【氏名又は名称】 インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

 【識別番号】 100086243

 【弁理士】

 【氏名又は名称】 坂口 博

【代理人】

 【識別番号】 100091568

 【弁理士】

 【氏名又は名称】 市位 嘉宏

【復代理人】

 【識別番号】 100079049

 【弁理士】

 【氏名又は名称】 中島 淳

 【電話番号】 03-3357-5171

【選任した復代理人】

 【識別番号】 100084995

 【弁理士】

【氏名又は名称】 加藤 和詳

【電話番号】 03-3357-5171

【選任した復代理人】

【識別番号】 100085279

【弁理士】

【氏名又は名称】 西元 勝一

【電話番号】 03-3357-5171

【選任した復代理人】

【識別番号】 100099025

【弁理士】

【氏名又は名称】 福田 浩志

【電話番号】 03-3357-5171

【手数料の表示】

【予納台帳番号】 006839

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 9304391

【包括委任状番号】 9304392

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 配列の変換方法、構造解析方法、装置、記録媒体及び伝送媒体

【特許請求の範囲】

【請求項 1】 複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と相補の関係を有する別の変数が存在している場合には、前記別の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記別の変数が存在していない場合には、前記別の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在している全ての変数について行って、前記処理対象の配列を変換する配列の変換方法。

【請求項 2】 複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と同一の変数が存在している場合には、前記同一の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記同一の変数が存在していない場合には、前記同一の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在する全ての変数について行って、前記処理対象の配列を第 1 の配列へ変換すると共に、

前記処理対象の配列を、請求項 1 記載の配列の変換方法により第 2 の配列へ変換し、

第 1 の配列及び第 2 の配列を用いて前記処理対象の配列の構造を解析する配列の構造解析方法。

【請求項 3】 前記第 1 の配列及び前記第 2 の配列を対応する一組の文字列とみなして単一の接尾辞木を作成すると共に、前記単一の接尾辞木の各枝にラベルとして付した前記第 1 の配列の部分配列及び前記第 2 の配列の部分配列のうち、前記各部分配列中に存在しない前記同一の変数又は前記別の変数の位置を指し示す情報を、前記同一の変数又は前記別の変数が存在していないことを表す情報に変換し、

作成した接尾辞木を用いて前記処理対象の配列の構造を解析する

ことを特徴とする請求項 2 記載の配列の構造解析方法。

【請求項 4】 前記別の変数又は前記同一の変数の位置を指し示す情報として、対象となる変数から前記同一の変数又は前記別の変数の位置までの前記処理対象の配列上での要素の数を表す数値情報を用いて、前記第 1 の配列及び前記第 2 の配列への変換を各々行い、

前記第 1 の配列及び第 2 の配列の一方に対し、該配列上に存在する前記別の変数又は前記同一の変数が存在しないことを表す情報を、他方の配列上の前記情報に対応する位置に存在する前記要素の数を表す数値情報の正負の符号を反転した数値情報に全て置き換え、

前記置き換えを行った配列を文字列とみなして接尾辞木を作成すると共に、前記接尾辞木の各枝にラベルとして付した前記置き換えを行った配列の部分配列のうち、前記各部分配列中に存在しない前記同一の変数又は前記別の変数の位置を指し示す数値情報を、前記同一の変数又は前記別の変数が存在していないことを表す情報に変換し、

作成した接尾辞木を用いて前記処理対象の配列の構造を解析する

ことを特徴とする請求項 2 記載の配列の構造解析方法。

【請求項 5】 前記作成した接尾辞木を用いて前記処理対象の配列中に頻出する同一構造の部分配列を抽出することで、前記処理対象の配列の構造を解析することを特徴とする請求項 3 又は請求項 4 記載の配列の構造解析方法。

【請求項 6】 前記処理対象の配列は、第 1 の処理対象の配列、第 1 の識別情報、前記第 1 の処理対象の配列との共通部分配列を検索すべき第 2 の処理対象の配列、及び第 2 の識別情報が順に並んだ配列であり、

前記作成した接尾辞木を用いて第 1 の処理対象の配列と第 2 の処理対象の配列との共通部分配列を検索することで、前記第 1 の処理対象の配列及び前記第 2 の処理対象の配列の構造を解析する

ことを特徴とする請求項 3 又は請求項 4 記載の配列の構造解析方法。

【請求項 7】 複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と同一の変数が存在している場合には

、前記同一の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記同一の変数が存在していない場合には、前記同一の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在する全ての変数について行って、前記処理対象の配列を第 1 の配列へ変換する第 1 の変換手段と、

複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と相補の関係を有する別の変数が存在している場合には、前記別の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記別の変数が存在していない場合には、前記別の変数が存在していないことを表す情報に変換することを、前記処理対象の配列中に存在している全ての変数について行って、前記処理対象の配列を第 2 の配列へ変換する第 2 の変換手段と、

前記第 1 の配列及び前記第 2 の配列を用いて前記処理対象の配列の構造を解析する解析手段と、

を含む配列の構造解析装置。

【請求項 8】 複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と同一の変数が存在している場合には、前記同一の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記同一の変数が存在していない場合には、前記同一の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在する全ての変数について行って、前記処理対象の配列を第 1 の配列へ変換する第 1 のステップ、

複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と相補の関係を有する別の変数が存在している場合には、前記別の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記別の変数が存在していない場合には、前記別の変数が存在していないことを表す情報に変換することを、前記処理対象の配列中に存在している全ての変数について行って、前記処理対象の配列を第 2 の配列へ変換する第 2 のステップ、

前記第 1 の配列及び前記第 2 の配列を用いて前記処理対象の配列の構造を解析

する第 3 のステップ

を含む処理をコンピュータに実行させるためのプログラムが記録された記録媒体。

【請求項 9】 複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と同一の変数が存在している場合には、前記同一の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記同一の変数が存在していない場合には、前記同一の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在する全ての変数について行って、前記処理対象の配列を第 1 の配列へ変換する第 1 のステップ、

複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と相補の関係を有する別の変数が存在している場合には、前記別の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記別の変数が存在していない場合には、前記別の変数が存在していないことを表す情報に変換することを、前記処理対象の配列中に存在している全ての変数について行って、前記処理対象の配列を第 2 の配列へ変換する第 2 のステップ、

前記第 1 の配列及び前記第 2 の配列を用いて前記処理対象の配列の構造を解析する第 3 のステップ

を含む処理をコンピュータに実行させるためのプログラムを伝送する伝送媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は配列の変換方法、構造解析方法、装置、記録媒体及び伝送媒体に係り、特に、配列の構造を解析するために配列を変換する配列の変換方法、前記配列の変換方法を利用して配列の構造を解析する配列の構造解析方法、前記配列の構造解析方法を適用可能な配列の構造解析装置、コンピュータによって前記配列の構造解析方法を実現するためのプログラムが記録された記録媒体、コンピュータ

によって前記配列の構造解析方法を実現するためのプログラムを伝送する伝送媒体に関する。

【0 0 0 2】

【従来の技術】

近年、様々な生物の遺伝情報全体が解明され、人間の遺伝情報全体の解明にも期待が寄せられている。遺伝情報は、DNAではアデニン（A）、チミン（T）、シトシン（C）、グアニン（G）の4種類の塩基の配列で表され、RNAではDNAにおけるTをウラシル（U）に代えた4種類の塩基の配列で表される。このため、遺伝情報の解析は、1本鎖DNAやRNAの塩基配列を便宜的に文字列へ置き換え、該文字列中の頻出部分文字列等のパターンを抽出し、抽出結果を分析することによって行われている。

【0 0 0 3】

文字列中に存在する頻出部分文字列や、2つ以上の文字列に共通な部分文字列等を高速で検索するのに有効な技術（データ構造）として、従来より接尾辞木（Suffix tree）が知られている。接尾辞木は、処理対象の文字列中に存在しない文字\$を処理対象の文字列の最後に加えた文字列における全ての接尾辞を表す木であり、一例として、処理対象の文字列「mississippi」の最後に文字\$を加えた文字列「mississippi\$」の接尾辞木は図7のようになる。

【0 0 0 4】

図7に示すように、接尾辞木は、各枝に部分文字列に相当するラベルが付されており、単一のノード（ルートノードを含む）から出て行く各枝に付されているラベルの最初の文字は全て異なり、それらはラベルの最初の文字に関してソートされている（例えば図7では、図の左側から右側へ向けて英語のアルファベット順に枝が並んでいる）。接尾辞木では、ルートノードから特定の葉ノード（他の枝が接続されていない枝の先端のノード）に至る各枝に付されたラベルを並べたものが、前記特定の葉ノードに対応する接尾辞となっている（例えば、「ppi\$」「ssi」「i」のラベルが付された各枝を経てルートノードに至る葉ノードに対応する接尾辞は「issippi\$」であり、「ssippi\$」「si」「s」のラベルが付された各枝を経てルートノードに至る葉ノードに対応する接尾辞は「ssissippi\$」であ

る)。

【0005】

接尾辞木については、接尾辞木のデータ構造を $O(n \log s)$ に相当する時間（但し、 n は元の文字列の長さ（文字数）、 s は元の文字列を構成するアルファベットの文字種数）で構成可能なアルゴリズムが知られており、特にアルファベットが整数アルファベット（1 から n までの数字）である場合は、接尾辞木のデータ構造を $O(n)$ に相当する時間で構成することができる。従って、処理対象の文字列が、DNA や RNA の塩基の配列を表す文字列等のように膨大な長さである場合にも、処理対象の文字列の接尾辞木のデータ構造を構成することを短い時間（詳しくは元の文字列の長さに対して線形な関係を有する時間）で完了させることができる。また、処理対象の文字列からの長さ（文字数） m の部分文字列の検索についても、接尾辞木を利用すれば $O(m \log s)$ に相当する時間で行うことができ、共通部分文字列や頻出部分文字列の列挙を短い時間（元の文字列の長さに対して線形な関係を有する時間）で行うことができる。

【0006】

また接尾辞木は、各枝に付したラベルを、該ラベルの文字列の最初の文字及び最後の文字（「\$」の前の文字）の、元の文字列における位置を表す情報に置き換える（例えば「mississippi\$」を [1:11] に置き換える）ことで、接尾辞木を表す文字列の長さを、元の文字列の長さの定数倍に収めることができるが、接尾辞木を表す文字列の長さを更に抑制するための技術として接尾辞配列 (Suffix Array) も知られている。

【0007】

前述のように、接尾辞木の葉ノードはそれぞれが一つずつ元の文字列の接尾辞に対応しているが、各接尾辞を、接尾辞木の一端側（例えば図7では左端側）の葉ノードに対応する接尾辞から順に（図7の辞書的順序参照）並べると、元の文字列の全ての接尾辞を辞書的順序で並べた配列が得られ、この配列の構成要素である各接尾辞を、接尾辞の最初の文字の元の文字列における位置を表す情報に置き換え（例えば「ippi\$」を [8] に置き換え）れば、元の文字列と同じ長さの配列（接尾辞配列という）が得られる。例えば図7の「mississippi」の接尾辞配列

は「8 5 2 11 1 10 9 7 4 6 3」となる。

【0 0 0 8】

上記の接尾辞配列を用いれば、接尾辞木を用いる場合と比較して文字列検索を行うために必要なメモリ容量を削減することができるが、文字列の検索時間は $O(m \log n)$ となる (n は処理対象の文字列の長さ、 m は検索したい文字列の長さ)。

【0 0 0 9】

また、文字列が変数を含む場合の頻出部分文字列や共通部分文字列を検索するための技術(データ構造)としてParameterized Suffix Treeも知られている。DNAやRNAの塩基配列等の生物配列では、配列中の特定の構成要素と他の特定の構成要素とが入れ替え可能な場合がある(例えばDNAのAとT、GとCは相補の関係を有し、入れ替え可能である)ことから、Parameterized Suffix Treeでは配列中の入れ替え可能な要素を変数として扱い、入れ替え可能な要素を含む配列に対応する文字列(変数を含む文字列)において、変数を入れ替えることで同一となり得る文字列を同じ文字列とみなしている。

【0 0 1 0】

例えば、 x, y, z を変数、 a, b, c を固定文字と考えたとき、「 $axbycxaza$ 」と「 $azbxczaya$ 」は、変数 x, y, z を互いに入れ替えることで同じ配列になるのと同じ文字列(p-string(Parameterized String)と称する)とみなされる。p-stringの検出には $prev()$ と表記されるエンコーディングが用いられる。これは文字列中の変数を、直前に出現した同一の変数との間の長さを表す数値(最初に出てきた変数は0)に置き換えるものであり、例えば前出の2つの文字列に対して $prev()$ によりエンコーディングを行うと、 $prev(axbycxaza) = prev(azbxczaya) = a0b0c4a0a$ と等しくなる。

【0 0 1 1】

Parameterized Suffix Tree は、処理対象の文字列に存在しない文字\$を加えた文字列の全ての接尾辞を $prev()$ によりエンコーディングしたものを表す木である(文字\$を加えた文字列を $prev()$ によりエンコードすることで得られる配列を通常の文字列だと考えて通常のsuffix treeを作ったものとは異なる)。接尾辞

木の場合と同様に、木の各葉ノードがそれぞれの接尾辞に対応している。各枝は部分文字列に相当するラベルを持ち、ルートノードから特定の葉ノードに至る各枝に付されたラベルを並べたものが、前記特定の葉ノードに対応する接尾辞を `prev()` でエンコーディングしたものになっている。

【0 0 1 2】

また、接尾辞木と同様に、単一のノード（ルートノードを含む）から出て行く各枝に付されているラベルの最初の文字は全て異なり、それらはラベルの最初の文字に関してソートされている。また、それぞれの枝のラベルは元の文字列における最初と最後の位置で表されるため、このデータ構造は文字列の長さに関して定数倍の大きさに収まる。

【0 0 1 3】

【発明が解決しようとする課題】

ところで、DNA や RNA の塩基配列等の生物配列においては、配列自体は全く異なるものの構造が同一の配列が、同じような機能や性質を有している可能性が高いことが知られており、例えば DNA の塩基配列において、相補の関係を有する A と T、G と C の何れか一方、又は両方が入れ替わっている場合や、相補の関係を有しない A と C が入れ替わりかつ相補の関係を有しない T と G が入れ替わっている場合は、元の配列と比較して配列自体は相違しているものの配列の構造（配列の構成要素相互の関係）は同一であり、上記の入れ替えを行うことで得られる配列の機能や性質は元の配列と似通っていることが多い。このように、生物配列の解析においては、配列自体が同一か否かに拘わらず配列の構造が同一の配列を同一の配列と見なして、頻出配列を抽出したり 2 つの配列に共通に含まれる部分配列を検索したりすることは、極めて重要な意味を持っている。

【0 0 1 4】

これに対し、先に述べた従来技術のうち、接尾辞木や接尾辞配列を利用した場合には、文字列自体が同一の文字列以外を同一の文字列として扱うことができないので、配列自体は異なるものの配列の構造が同一の配列を同一の配列として扱うことができないという欠点がある。また Parameterized Suffix Tree は、単に変数が入れ替わっている文字列を同一の文字列として扱うので、例えば DNA の

塩基配列において、AとCのみが入れ替わっている場合やTとGのみが入れ替わっている場合、或いはA→C→T→Aと入れ替わっている等のように元の配列と構造が異なる配列と、元の配列と構造が同一の配列を区別することができない。従って、上述した従来技術の何れを用いたとしても、生物配列の解析を効率良く行うことは困難であった。

【0015】

本発明は上記事実を考慮して成されたもので、配列の構造を効率良く解析できるように配列を変換することができる配列の変換方法を得ることが目的である。

【0016】

また本発明は、配列の構造解析を効率良く行うことができる配列の構造解析方法、配列の構造解析装置、記録媒体及び伝送媒体を得ることが目的である。

【0017】

【課題を解決するための手段】

上記目的を達成するために本発明に係る配列の変換方法は、複数種の構成要素（各構成要素の全てが他の構成要素へ置換可能な変数であってもよいし、変数以外の他の構成要素が含まれていてもよい）の組み合わせから成る配列中に存在している変数を、前記配列を一定方向（例えば配列の一端から他端へ向かう方向、又はその逆の方向）から見たときに前記変数よりも上流側に相当する位置に前記変数と相補の関係を有する別の変数が存在している場合には、前記別の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記別の変数が存在していない場合には、前記別の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在している全ての変数について行う。

【0018】

上記の変換により、配列中の相補の関係を有する変数同士が、互いの位置関係を表す情報（互いの存在の有無及び位置を表す情報）に変換されるので、相補の関係を有する変数の組が複数存在し、かつ異なる変数の組の間で変数を入れ替えた配列（すなわち配列の構造（配列の構成要素相互の関係）は等しいものの配列自体は異なる配列）は、本発明に係る配列の変換方法により互いに等しい配列に変換される。

【 0 0 1 9 】

一例として、変数 x と変数 z が相補の関係を有し、変数 y と変数 w が相補の関係を有している場合、配列 $(A B x B y A z w z)$ は、本発明に係る配列の変換方法（以下では $\text{compl}()$ と表記）により、例えば

$$\text{compl}(A B x B y A z w z) = A B 0 B 0 A 4 3 6$$

と変換される（変換後の配列中の「0」は相補の関係を有する変数が上流側に存在していないことを意味し、「4」「3」「6」の各数値は上流側に存在する相補の関係を有する変数との距離を表している）。

【 0 0 2 0 】

上記の元の配列に対し、構造が同一の配列として、(1)相補の関係を有する変数 x と変数 z のみを入れ替えた配列 $(A B z B y A x w x)$ 、(2)相補の関係を有する変数 y と変数 w のみを入れ替えた配列 $(A B x B w A z y z)$ 、(3)相補の関係を有しない変数 x と変数 w 、変数 y と変数 z を各々入れ替えた配列 $(A B w B z A y x y)$ を、本発明に係る配列の変換方法によって各々変換すると、

$$(1) \text{compl}(A B z B y A x w x) = A B 0 B 0 A 4 3 6$$

$$(2) \text{compl}(A B x B w A z y z) = A B 0 B 0 A 4 3 6$$

$$(3) \text{compl}(A B w B z A y x y) = A B 0 B 0 A 4 3 6$$

となり、元の配列と等しい配列に変換される。

【 0 0 2 1 】

一方、元の配列と構造が異なる配列の一例として、相補の関係を有しない変数 x と変数 w のみを入れ替えた配列 $(A B w B y A z x z)$ を本発明に係る配列の変換方法によって変換すると、

$$\text{compl}(A B w B y A z x z) = A B 0 B 2 A 0 1 1$$

となり、元の配列と異なる配列に変換される。

【 0 0 2 2 】

上記より明らかなように、例えば一对の配列を本発明に係る配列の変換方法によって各々変換すれば、変換後の一对の配列を比較することで、一对の配列の構造が同一か否かを判断したり、一对の配列の双方に構造が同一の部分配列が含まれているか否かを判断することを効率良く行うことができる。また、或る配列を

本発明に係る配列の変換方法によって変換すれば、変換後の配列に基づいて前記配列中に頻出する同一構造の部分配列の抽出も容易に判断することができる。従って、本発明に係る配列の変換方法によれば、配列の構造を効率良く解析できるように配列を変換することができる。

【0023】

ところで、本発明に係る配列の変換方法によれば、構造が同一の配列を等しい配列へ変換することができるが、構造が異なる配列が等しい配列へ変換される例外ケースも僅かではあるが存在している。例えばDNAの塩基配列において、配列(T T A A)と配列(A G C C)は構造が互いに異なっているが、これらの配列を本発明に係る配列の変換方法によって変換すると、

$\text{compl}(\text{T T A A}) = \text{compl}(\text{A G C C}) = (0 0 1 2)$

と互いに等しい配列に変換される。

【0024】

このため本発明に係る配列の構造解析方法は、複数種の構成要素の組み合わせから成る配列中に存在している他の構成要素へ置換可能な変数を、前記配列を一定方向から見たときに前記変数よりも上流側に相当する位置に前記変数と同一の変数が存在している場合には、前記同一の変数の位置を指し示す情報に変換し、前記上流側に相当する位置に前記同一の変数が存在していない場合には、前記同一の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在する全ての変数について行って、前記処理対象の配列を第1の配列へ変換している。

【0025】

上記の第1の配列への変換により、配列中の変数は、配列中の同一変数の位置関係を表す情報(同一の変数の存在の有無及び位置を表す情報)に変換されるので、例えば構造が異なっているものの本発明に係る配列の変換方法では等しい配列へ変換される一対の配列を、第1の配列として異なる配列に変換することができる。例えば本発明に係る配列の変換方法では等しい配列に変換される配列(T T A A)及び配列(A G C C)は、上記の変換方法(以下ではprev()と表記)により第1の配列に変換すると、

$\text{prev}(\text{T T A A}) = (0 1 0 1)$ $\text{prev}(\text{A G C C}) = (0 0 0 1)$

と異なる配列に変換される（変換後の第 1 の配列中の「0」は同一の変数が上流側に存在していないことを意味し、「4」「2」の各数値は上流側に存在する同一の変数との距離を表している）。

【0 0 2 6】

本発明に係る配列の構造解析方法では、処理対象の配列を、上記のようにして第 1 の配列に変換すると共に、本発明に係る配列の変換方法により第 2 の配列へ変換し、第 1 の配列及び第 2 の配列を用いて処理対象の配列の構造を解析するので、例えば変換後の第 1 の配列及び第 2 の配列が各々等しい配列を構造が同一の配列と判断する等により、構造が異なる配列を除外して同一の構造の配列のみを確実に判断することができ、配列の構造解析を効率良く行うことができる。

【0 0 2 7】

ところで、第 1 の配列及び第 2 の配列を用いて配列の構造を解析する際の解析方法としては、種々の解析方法が考えられるが、例えば第 1 の配列及び第 2 の配列を用いて接尾辞木を作成し、作成した接尾辞木を用いて配列の構造を解析するようにすれば、他の解析方法と比較して処理が短時間で完了するので好ましい。接尾辞木の作成は、例えば、第 1 の配列及び第 2 の配列を対応する一組の文字列とみなして単一の接尾辞木を作成すると共に、前記単一の接尾辞木の各枝にラベルとして付した第 1 の配列の部分配列及び第 2 の配列の部分配列のうち、各部分配列中に存在しない前記同一の変数又は前記別の変数の位置を指し示す情報を、前記同一の変数又は前記別の変数が存在していないことを表す情報に変換することで行うことができる。

【0 0 2 8】

また、第 1 の配列における同一の変数の位置を指し示す情報、及び第 2 の配列における別の変数の位置を指し示す情報として、対象となる変数から同一の変数又は別の変数の位置までの処理対象の配列上での要素の数を表す数値情報を用いて、処理対象の配列を第 1 の配列及び第 2 の配列へ変換する場合、一対の配列を第 1 の配列及び第 2 の配列へ変換したときに、第 1 の配列及び第 2 の配列の一方（配列 A とする）が一致すると共に、他方の配列（配列 B とする）のうち、少な

くとも配列 A 上の別の変数又は同一の変数が存在しないことを表す情報と対応する位置に存在する数値情報が一致することがわかっていれば、一对の配列から得られる配列 B も一致し、一对の配列は構造が同一の配列であると判断できる。

【 0 0 2 9 】

上記に基づき、変換によって得られた第 1 の配列及び第 2 の配列の一方に対し、該配列上に存在する前記別の変数又は前記同一の変数が存在しないことを表す情報を、他方の配列上の前記情報に対応する位置に存在する前記要素の数を表す数値情報の正負の符号を反転した数値情報に全て置き換え、前記置き換えを行った配列を文字列とみなして接尾辞木を作成すると共に、前記接尾辞木の各枝にラベルとして付した前記置き換えを行った配列の部分配列のうち、前記各部分配列中に存在しない前記同一の変数又は前記別の変数の位置を指し示す数値情報を、前記同一の変数又は前記別の変数が存在していないことを表す情報に変換することで接尾辞木を作成するようにしてもよい。

【 0 0 3 0 】

また、上記のように接尾辞木を作成した場合、処理対象の配列の構造解析の一種である、処理対象の配列中に頻出する同一構造の部分配列を抽出することは、作成した接尾辞木を用いることで極めて容易に行うことができる。

【 0 0 3 1 】

また、処理対象の配列の構造解析の一種である、第 1 の処理対象の配列と第 2 の処理対象の配列との共通部分配列を検索することについても、処理対象の配列（接尾辞の作成対象の配列）として、第 1 の処理対象の配列、第 1 の識別情報、第 2 の処理対象の配列、及び第 2 の識別情報が順に並んだ配列を用いて接尾辞木を作成し、作成した接尾辞木を用いることで、前記共通部分配列の検索を極めて容易に行うことができる。

【 0 0 3 2 】

なお、接尾辞木は文字列を一定の方向から見たときに、文字列中に存在する全ての接尾辞を木構造で表現したものであるが、文字列中の接尾辞は前記一定の方向と逆の方向から見れば接頭辞であり、配列の構造解析に用いることを前提とすれば、文字列中に存在する全ての接頭辞を木構造で表現した接頭辞木も、論理的

には接尾辞木と等価である。従って、本発明に係る接尾辞木として接頭辞木を用いてもよいことは言うまでもない。

【 0 0 3 3 】

本発明に係る配列の構造解析装置は、配列中に存在している他の構成要素へ置換可能な変数を、同一の変数の位置を指し示す情報又は同一の変数が存在していないことを表す情報に変換することで処理対象の配列を第 1 の配列へ変換する第 1 の変換手段と、配列中に存在している他の構成要素へ置換可能な変数を、前記変数と相補の関係を有する別の変数の位置を指し示す情報又は前記別の変数が存在していないことを表す情報に変換することで処理対象の配列を第 2 の配列へ変換する第 2 の変換手段と、第 1 の配列及び第 2 の配列を用いて処理対象の配列の構造を解析する解析手段と、を含んで構成されているので、本発明に係る配列の構造解析方法と同様に、配列の構造解析を効率良く行うことができる。

【 0 0 3 4 】

本発明に係る記録媒体には、配列中に存在している他の構成要素へ置換可能な変数を、同一の変数の位置を指し示す情報又は同一の変数が存在していないことを表す情報に変換することで処理対象の配列を第 1 の配列へ変換する第 1 のステップ、配列中に存在している他の構成要素へ置換可能な変数を、前記変数と相補の関係を有する別の変数の位置を指し示す情報又は前記別の変数が存在していないことを表す情報に変換することで処理対象の配列を第 2 の配列へ変換する第 2 のステップ、第 1 の配列及び第 2 の配列を用いて処理対象の配列の構造を解析する第 3 のステップを含む処理、すなわち本発明に係る配列の構造解析方法をコンピュータによって実現するためのプログラムが記録されているので、コンピュータが前記記録媒体に記録されたプログラムを読み出して実行することにより、本発明に係る配列の構造解析方法と同様に、配列の構造解析を効率良く行うことができる。

【 0 0 3 5 】

本発明に係る伝送媒体は、配列中に存在している他の構成要素へ置換可能な変数を、同一の変数の位置を指し示す情報又は同一の変数が存在していないことを表す情報に変換することで処理対象の配列を第 1 の配列へ変換する第 1 のステッ

ブ、配列中に存在している他の構成要素へ置換可能な変数を、前記変数と相補の関係を有する別の変数の位置を指し示す情報又は前記別の変数が存在していないことを表す情報に変換することで処理対象の配列を第 2 の配列へ変換する第 2 のステップ、第 1 の配列及び第 2 の配列を用いて処理対象の配列の構造を解析する第 3 のステップを含む処理、すなわち本発明に係る配列の構造解析方法をコンピュータによって実現するためのプログラムを伝送するので、コンピュータが前記伝送媒体によって伝送されたプログラムを記憶手段に一時記憶した後に前記記憶手段から読み出して実行することにより、本発明に係る配列の構造解析方法と同様に、配列の構造解析を効率良く行うことができる。

【 0 0 3 6 】

【発明の実施の形態】

以下、図面を参照して本発明の実施形態の一例を詳細に説明する。図 1 には、本発明を実現するのに適した典型的なパーソナル・コンピュータ（PC）から成るコンピュータ・システム 10 のハードウェア構成がサブシステム毎に模式的に示されている。本発明を実現する PC の一例は、O A D G（PC Open Architecture Developer's Group）仕様に準拠し、オペレーティング・システム（OS）として米マイクロソフト社の” Windows 9 8 又は NT ” 又は米 IBM 社の” OS / 2 ” を搭載した PC（ノートブック型でもデスクトップ型でもよい）である。以下、コンピュータ・システム 10 の各部について説明する。

【 0 0 3 7 】

コンピュータ・システム 10 全体の頭脳である CPU 14 は、OS の制御下で、各種プログラムを実行する。CPU 14 は、例えば米インテル社製の CPU チップ” Pentium ”、” MMX テクノロジ Pentium ”、” Pentium Pro ” や、AMD 社等の他社製の CPU でも良いし、IBM 社製の” PowerPC ” でも良い。CPU 14 は、頻繁にアクセスするごく限られたコードやデータを一時格納することで、メイン・メモリ 16 への総アクセス時間を短縮するための高速動作メモリである L 2（レベル 2）－キャッシュを含んで構成されている。L 2－キャッシュは、一般に SRAM（スタティック RAM）チップで構成され、その記憶容量は例えば 5 1 2 k B 又はそれ以上である。

【 0 0 3 8 】

CPU 1 4 は、自身の外部ピンに直結されたプロセッサ直結バスとしての FSB 1 8、高速の I/O 装置用バスとしての PCI (Peripheral Component Interconnect) バス 2 0、及び低速の I/O 装置用バスとしての ISA (Industry Standard Architecture) バス等から成る I/O バス 2 2 という 3 階層のバスを介して、後述の各ハードウェア構成要素と相互接続されている。

【 0 0 3 9 】

FSB 1 8 と PCI バス 2 0 は、一般にメモリ/PCI 制御チップ 2 4 と呼ばれるブリッジ回路 (ホスト-PCI ブリッジ) によって連絡されている。本実施形態のメモリ/PCI 制御チップ 2 4 は、メイン・メモリ 1 6 へのアクセス動作を制御するためのメモリ・コントローラ機能や、FSB 1 8 と PCI バス 2 0 の間のデータ転送速度の差を吸収するためのデータ・バッファ等を含んだ構成となっており、例えばインテル社製の 4 4 0 EX や 4 4 0 GX 等を用いることができる。

【 0 0 4 0 】

メイン・メモリ 1 6 は、CPU 1 4 の実行プログラムの読み込み領域として、或いは実行プログラムの処理データを書き込む作業領域として利用される書き込み可能メモリである。メイン・メモリ 1 6 は、一般には複数個の DRAM (ダイナミック RAM) チップで構成され、例えば 3 2 MB を標準装備し 2 5 6 MB まで増設可能である。近年では、更に高速化の要求に応えるべく、DRAM は高速ページ DRAM、EDO DRAM、シンクロナス DRAM (SDRAM)、バースト EDO DRAM、RDRAM 等へと変遷している。

【 0 0 4 1 】

なお、ここでいう実行プログラムには、Windows 9 8 等の OS、周辺機器類をハードウェア操作するための各種デバイス・ドライバ、特定業務に向けられたアプリケーション・プログラムや、フラッシュ ROM 5 6 (詳細は後述) に格納された BIOS (Basic Input/Output System: キーボードやフロッピーディスク・ドライブ等の各ハードウェアの入出力操作を制御するためのプログラム) 等のファームウェアが含まれる。

【 0 0 4 2 】

P C I バス 2 0 は、比較的高速なデータ伝送が可能なタイプのバス（例えばバス幅 3 2 / 6 4 ビット、最大動作周波数 3 3 / 6 6 / 1 0 0 M H z、最大データ転送速度 1 3 2 / 2 6 4 M B p s）であり、カードバス・コントローラ 3 0 のような比較的高速で駆動する P C I デバイス類がこれに接続される。なお、P C I アーキテクチャは、米インテル社の提唱に端を発したものであり、いわゆる P n P（プラグ・アンド・プレイ）機能を実現している。

【 0 0 4 3 】

ビデオ・サブシステム 2 6 は、ビデオに関連する機能を実現するためのサブシステムであり、C P U 1 4 からの描画命令を実際に処理し、処理した描画情報をビデオメモリ（V R A M）に一旦書き込むと共に、V R A M から描画情報を読み出して液晶ディスプレイ（L C D）に描画データとして出力するビデオ・コントローラを含む。また、ビデオ・コントローラは、付設されたデジタル－アナログ変換器（D A C）によってデジタルのビデオ信号をアナログのビデオ信号へ変換することができる。アナログのビデオ信号は、信号線を介して C R T ポート（図示省略）へ出力される。

【 0 0 4 4 】

また、P C I バス 2 0 にはカードバス・コントローラ 3 0、オーディオ・サブシステム 3 2 及びモデム・サブシステム 3 4 が各々接続されている。カードバス・コントローラ 3 0 は、P C I バス 2 0 のバス・シグナルを P C I カードバス・スロット 3 6 のインタフェース・コネクタ（カードバス）に直結させるための専用コントローラである。カードバス・スロット 3 6 には、例えば P C 本体の壁面に配設され、P C M C I A（Personal Computer Memory Association）／J E I D A（Japan Electronic Industry Development Association）が策定した仕様（例えば” P C Card standard 95”）に準拠した P C カード（図示せず）が装填される。

【 0 0 4 5 】

また、モデム・サブシステム 3 4 には L A N や電話回線等の通信回線が接続される。コンピュータ・システム 1 0 はこれらの通信回線を介してインターネット

に接続可能とされている。

【0046】

PCIバス20とI/Oバス22は多機能PCIデバイス38によって相互に接続されている。多機能PCIデバイス38は、PCIバス20とI/Oバス22とのブリッジ機能、DMAコントローラ機能、プログラマブル割り込みコントローラ(PIC)機能、及びプログラマブル・インターバル・タイマ(PIT)機能、IDE(Integrated Drive Electronics)インタフェース機能、USB(Universal Serial Bus)機能、SMB(System Management Bus)インタフェース機能を備えており、例えばインテル社製のPIIX4というデバイスを用いることができる。

【0047】

なお、DMAコントローラ機能は、周辺機器(たとえばFDD)とメイン・メモリ16との間のデータ転送をCPU14の介在なしに実行するための機能である。またPIC機能は、周辺機器からの割り込み要求(IRQ)に応答して所定のプログラム(割り込みハンドラ)を実行させる機能である。また、PIT機能はタイマ信号を所定周期で発生させる機能であり、その発生周期はプログラマブルである。

【0048】

また、IDEインタフェース機能によって実現されるIDEインタフェースには、IDEハードディスク・ドライブ(HDD)40が接続される他、IDE CD-ROMドライブ42がATAPI(AT Attachment Packet Interface)接続される。また、IDE CD-ROMドライブ42の代わりに、DVD(Digital Video Disc又はDigital Versatile Disc)ドライブのような他のタイプのIDE装置が接続されていても良い。HDD40やCD-ROMドライブ42等の外部記憶装置は、例えばPC本体内の「メディア・ベイ」又は「デバイス・ベイ」と呼ばれる収納場所に格納される。これら標準装備された外部記憶装置は、FDDやバッテリー・パックのような他の機器類と交換可能かつ排他的に取り付けられる場合もある。

【0049】

なお、メイン・メモリ 1 6 は本発明に係る主記憶装置に対応しており、HDD 4 0 は本発明に係る二次記憶装置に対応している。

【0 0 5 0】

また、多機能PCIデバイス38にはUSBポートが設けられており、このUSBポートは、例えばPC本体の壁面等に設けられたUSBコネクタ44と接続されている。USBは、電源投入のまま新しい周辺機器（USBデバイス）を抜き差しする機能（ホット・プラグング機能）や、新たに接続された周辺機器を自動認識しシステム・コンフィギュレーションを再設定する機能（プラグ・アンド・プレイ）機能をサポートしている。1つのUSBポートに対して、最大63個のUSBデバイスをディジーチェーン接続することができる。USBデバイスの例は、キーボード、マウス、ジョイスティック、スキャナ、プリンタ、モデム、ディスプレイモニタ、タブレットなど様々である。

【0 0 5 1】

更に、多機能PCIデバイス38にはSMバスを介してEEPROM（図示省略）が接続されている。EEPROMはユーザによって登録されたパスワードやスーパーバイザー・パスワード、製品シリアル番号等の情報を保持するためのメモリであり、不揮発性で記憶内容を電氣的に書き替え可能とされている。

【0 0 5 2】

I/Oバス22は、PCIバス20よりもデータ転送速度が低いバスであり（例えばバス幅16ビット、最大データ転送速度4Mbps）、Super I/Oコントローラ46、電源コントローラ48、EEPROM等から成るフラッシュROM56、CMOS58に加え、リアルタイム・クロック（RTC）や、キーボード/マウス・コントローラのような比較的低速で動作する周辺機器類（何れも図示省略）を接続するのに用いられる。

【0 0 5 3】

Super I/Oコントローラ46にはI/Oポート52が接続されており、フロッピーディスク・ドライブ（FDD）の駆動、パラレル・ポートを介したパラレル・データの入出力（PIO）、シリアル・ポートを介したシリアル・データの入出力（SIO）を制御するための周辺コントローラである。

【 0 0 5 4 】

電源コントローラ 4 8 は主にコンピュータ・システム 1 0 のパワー・マネージメントやサーマル・マネージメントを行うものであり、MPU, RAM, ROM 及びタイマ等を備えたシングルチップ・マイコンで構成することができる。ROM にはパワー・マネージメントやサーマル・マネージメントを実行するのに必要なプログラム及び参照テーブルが格納されている。電源コントローラ 4 8 にはパワー・サプライ・コントローラ 5 4 が接続されている。パワー・サプライ・コントローラ 5 4 には、バッテリーを充電するための充電器、コンピュータ・システム 1 0 で使用される 5 V, 3. 3 V 等の直流定電圧を生成するための DC / DC コンバータが含まれ、電源コントローラ 4 8 の下で電力制御を行う。

【 0 0 5 5 】

フラッシュ ROM 5 6 は、BIOS やブート・ストラップ・コード等のファームウェアのプログラムを保持するためのメモリであり、不揮発性で記憶内容を電氣的に書き替え可能とされている。また、CMOS 5 8 は揮発性の半導体メモリがバックアップ電源に接続されて構成されており、不揮発性でかつ高速の記憶手段として機能する。

【 0 0 5 6 】

なお、コンピュータ・システム 1 0 を構成するためには、図 1 に示した以外にも多くの電気回路が必要である。但し、これらは当業者には周知であり、また、本発明の要旨を構成するものではないので、本明細書中では説明を省略する。また、図面の錯綜を回避するため、図中の各ハードウェアブロック間の接続も一部しか図示していないことを付記しておく。

【 0 0 5 7 】

次に本実施形態の作用を説明する。本実施形態では、本発明に係る配列の変換方法及び配列の構造解析方法が配列の構造解析プログラムによって実現される。配列の構造解析プログラムをコンピュータシステム 1 0 にインストール（移入）するには幾つかの方法があるが、例えば配列の構造解析プログラムをインストールするためのセットアッププログラムを配列の構造解析プログラム本体と共にフロッピーディスク等の情報記憶媒体 6 0 （図 1 参照）に記録しておき、この情報

記憶媒体 6 0 をコンピュータシステム 1 0 の I / O ポート 5 2 に接続された F D D にセットし、C P U 1 4 に対して前記セットアッププログラムの実行を指示すれば、情報記憶媒体 6 0 から配列の構造解析プログラムが順に読み出され、読み出された配列の構造解析プログラムが H D D 4 0 に順に書き込まれることで、配列の構造解析プログラムのインストールが行われる。

【 0 0 5 8 】

インストールされた配列の構造解析プログラムは、電源が投入されて既に稼動状態にあるコンピュータシステム 1 0 に対して、構造接尾辞木生成処理の実行が指示されると、C P U 1 4 によって H D D 4 0 から配列の構造解析プログラムが読み出されて実行される。これにより、コンピュータシステム 1 0 は本発明に係る配列の構造解析装置として機能する。このように、情報記憶媒体 6 0 は本発明に係る記録媒体に対応している。

【 0 0 5 9 】

次に、C P U 1 4 が配列の構造解析プログラムの一部である構造接尾辞木生成プログラムを実行することによって実現される構造接尾辞木生成処理について、図 2 のフローチャートを参照して説明する。

【 0 0 6 0 】

ステップ 2 0 0 では処理対象（構造解析対象）の文字列 S （処理対象の配列）を取り込む。なお、処理対象の文字列 S としては、例えば 4 種類の塩基（アデニン、チミン、シトシン、グアニン）を各々「A」「T」「C」「G」の文字に置き換えて一本鎖 D N A の塩基配列を表す文字列や、4 種類の塩基（アデニン、ウラシル、シトシン、グアニン）を各々「A」「U」「C」「G」の文字に置き換えて R N A の塩基配列を表す文字列を用いることができる。また、ステップ 2 0 0 で取り込む文字列 S には、文字列の末尾であることを表す末尾識別文字（文字列中に存在しない文字、例えば「\$」）が付加されている。

【 0 0 6 1 】

なお、以下では文字列 S の i 番目の文字を S [i]、文字列 S の j 番目の文字から始まって i 番目の文字で終わる部分文字列を S [j..i]、文字列 S の長さ（文字数）を n （すなわち S [1..n] = 文字列 S となる）で表す。

【 0 0 6 2 】

次のステップ 2 0 2 では、処理対象の文字列 S を変換条件 $\text{prev}()$ に従って第 1 の文字列 S_1 へ変換する ($\text{prev}(S) = S_1$) と共に、変換条件 $\text{compl}()$ に従って第 2 の文字列 S_2 へ変換する。処理対象の文字列 S の第 1 の文字列 S_1 への変換は、例えば図 3 に示す $\text{prev}()$ 演算処理によって行うことができる。この $\text{prev}()$ 演算処理は、本発明に係る配列の構造解析方法における第 1 の配列への変換に対応しており、以下、この処理について説明する。

【 0 0 6 3 】

ステップ 1 3 0 では文字位置レジスタ X_1, X_2, \dots に 0 を設定する。なお、文字位置レジスタは処理対象の文字列 S に含まれる変数（文字列 S の構造の解析において他の文字（変数）へ置換可能な文字：例えば一本鎖 DNA の塩基配列を表す文字列では「A」「T」「C」「G」、RNA の塩基配列を表す文字列では「A」「U」「C」「G」）の種類数と同数設けられている。また、次のステップ 1 3 2 ではカウンタ i に「1」を代入する。

【 0 0 6 4 】

ステップ 1 3 4 では文字列 S から i 番目の文字 $S[i]$ を取り出し、ステップ 1 3 6 で文字 $S[i]$ が変数か否か判定する。前記判定が否定された場合（例えば文字列 S が DNA や RNA の塩基配列を表す文字列で文字 $S[i]$ が末尾識別文字であった場合、或いは文字列 S が変数以外の文字も含む文字列で文字 $S[i]$ が変数以外の文字であった場合）にはステップ 1 3 8 へ移行し、第 1 の文字列 S_1 の i 番目の文字として文字 $S[i]$ そのものを記憶し、ステップ 1 5 0 へ移行する。

【 0 0 6 5 】

また、文字 $S[i]$ が変数の場合には、ステップ 1 3 6 の判定が肯定されてステップ 1 4 0 へ移行し、文字（変数） $S[i]$ の変数種 α を判別する。そしてステップ 1 4 2 では、文字（変数） $S[i]$ の変数種 α に対応する文字位置レジスタ X_α に設定されている数値が「0」か否か判定する。文字位置レジスタ X_α に設定されている数値が「0」であった場合、文字 $S[i]$ は最初に出現した変数種 α の変数であり、文字列 S 上の文字 $S[i]$ よりも上流側には同一の変数（同一の変数種 α の変数）は存在していないと判断できる。

【0066】

このため、ステップ142の判定が肯定された場合にはステップ144へ移行し、第1の文字列S1のi番目の文字として「0」（文字列S上の文字S[i]よりも上流側には同一の変数が存在していないことを表す情報）を記憶し、ステップ148へ移行する。ステップ148では文字位置レジスタX α にカウンタiの値（文字列S上の文字S[i]の位置を表す情報）を代入する。これにより、次に変数種 α の変数が出現したときにはステップ142の判定が肯定されることになる。

【0067】

次のステップ152ではカウンタiの値が文字列Sの長さ（文字数）nに一致したか否か判定する。判定が否定された場合には、ステップ152でカウンタiの値を「1」だけインクリメントした後にステップ134へ戻り、文字S[i]として次の文字を取り出してステップ136以降を繰り返す。

【0068】

また、文字S[i]が変数であり、文字S[i]と同一の変数が以前に出現している場合には、ステップ136の判定が肯定されると共に、ステップ142の判定が否定されてステップ146へ移行する。ステップ146では、第1の文字列S1のi番目の文字として「i-X α 」を記憶し、ステップ148へ移行する。このとき、文字位置レジスタX α には前回出現した文字S[i]と同一の変数の文字列S上での位置が記憶されているので、「i-X α 」は、文字列S上の文字S[i]よりも上流側に相当する位置に存在している文字S[i]と同一の変数の位置を指し示す（より詳しくは文字S[i]と前回出現した同一の変数との隔たりを表す）数値情報である。

【0069】

上述したprev()演算処理により、文字列S中に存在する全ての変数は、変数よりも上流側に同一の変数が存在している場合には、該同一の変数との隔たりを表す数値に変換され、変数よりも上流側に同一の変数が存在していない場合には「0」に変換される。これにより、例えばRNAの塩基配列を表す文字列S(AUAUCGU\$)は、第1の文字列S1として以下の文字列に変換されることになる。

$\text{prev}(\text{AUAUCGU\$}) = S1 = (0022003\$)$

【 0 0 7 0 】

また、処理対象の文字列 S の第 2 の文字列 $S2$ への変換は、例えば図 4 に示す $\text{compl}()$ 演算処理によって行うことができる。この $\text{compl}()$ 演算処理は、本発明に係る配列の変換方法、及び本発明に係る配列の構造解析方法における第 2 の配列への変換に対応している。以下、この $\text{compl}()$ 演算処理について説明する。

【 0 0 7 1 】

$\text{compl}()$ 演算処理では処理対象の文字列 S に含まれる変数に対し、相補の関係にある変数の組が予め定められている。例えば一本鎖 DNA の塩基配列を表す文字列 S に対しては、実際の DNA における各塩基の関係に基づき、相補の関係にある変数の組として A と T、C と G の 2 つの組が予め定められており、RNA の塩基配列を表す文字列に対しては、実際の RNA における各塩基の関係に基づき、相補の関係にある変数の組として A と U、C と G の 2 つの組が予め定められている。

【 0 0 7 2 】

ステップ 1 6 0 では文字位置レジスタ X_1, X_2, \dots に 0 を設定し、ステップ 1 6 2 ではカウンタ i に「1」を代入する。ステップ 1 6 4 では文字列 S から i 番目の文字 $S[i]$ を取り出し、ステップ 1 6 6 で文字 $S[i]$ が変数か否か判定する。前記判定が否定された場合にはステップ 1 6 8 へ移行し、第 2 の文字列 $S2$ の i 番目の文字として文字 $S[i]$ そのものを記憶し、ステップ 1 8 0 へ移行する。

【 0 0 7 3 】

また、文字 $S[i]$ が変数の場合には、ステップ 1 6 6 の判定が肯定されてステップ 1 7 0 へ移行し、文字（変数） $S[i]$ の変数種 α を判別すると共に、変数種 α の変数と相補の関係にある別の変数（例えば DNA の塩基配列を表す文字列 S において、文字 $S[i]$ が「A」であれば「T」、文字 $S[i]$ が「T」であれば「A」、文字 $S[i]$ が「C」であれば「G」、文字 $S[i]$ が「G」であれば「C」）の変数種 β を判別する。そしてステップ 1 7 2 では、文字（変数） $S[i]$ と相補の関係にある変数種 β の文字位置レジスタ X_β に設定されている数値が「0」か否か判定する。文字位置レジスタ X_β に設定されている数値が「0」であった場合

、変数種 β の変数は未だ出現しておらず、文字列 S 上の文字 $S[i]$ よりも上流側には文字 $S[i]$ の変数と相補の関係にある別の変数（変数種 β の変数）は存在していないと判断できる。

【0074】

このため、ステップ 172 の判定が肯定された場合にはステップ 174 へ移行し、第 2 の文字列 S_2 の i 番目の文字として「0」（文字列 S 上の文字 $S[i]$ よりも上流側には相補の関係を有する別の変数が存在していないことを表す情報）を記憶し、ステップ 178 へ移行する。ステップ 178 では文字位置レジスタ X_α にカウンタ i の値（文字列 S 上の文字 $S[i]$ の位置を表す情報）を代入する。これにより、相補の関係を有する変数種 β の変数が出現したときにはステップ 172 の判定が肯定されることになる。

【0075】

次のステップ 182 ではカウンタ i の値が文字列 S の長さ（文字数） n に一致したか否か判定する。判定が否定された場合には、ステップ 182 でカウンタ i の値を「1」だけインクリメントした後にステップ 164 へ戻り、文字 $S[i]$ として次の文字を取り出してステップ 166 以降を繰り返す。

【0076】

また、文字 $S[i]$ が変数であり、文字 $S[i]$ と相補の関係を有する別の変数が以前に出現している場合には、ステップ 166 の判定が肯定されると共に、ステップ 172 の判定が否定されてステップ 176 へ移行する。ステップ 176 では、第 2 の文字列 S_2 の i 番目の文字として「 $i - X_\beta$ 」を記憶し、ステップ 178 へ移行する。このとき、文字位置レジスタ X_β には前回出現した文字（変数） $S[i]$ と相補の関係を有する別の変数の文字列 S 上での位置が記憶されているので、「 $i - X_\beta$ 」は、文字列 S 上の文字 $S[i]$ よりも上流側に相当する位置に存在している文字 $S[i]$ と相補の関係を有する別の変数の位置を指し示す（より詳しくは文字 $S[i]$ と前回出現した相補の関係を有する変数との隔たりを表す）数値情報である。

【0077】

上述した `compl()` 演算処理により、文字列 S 中に存在する全ての変数は、変数

よりも上流側に相補の関係を有する変数が存在している場合には、該相補の関係を有する変数との隔たりを表す数値に変換され、変数よりも上流側に前記相補の関係を有する変数が存在していない場合には「0」に変換される。これにより、例えばRNAの塩基配列を表す文字列S (AUAUCGU\$)は、第2の文字列S2として以下の文字列に変換されることになる。

$\text{compl}(\text{AUAUCGU\$}) = \text{S2} = (0111014\$)$

【0078】

上記のようにして、処理対象の文字列Sを第1の文字列S1及び第2の文字列S2へ各々変換すると、図2のフローチャートのステップ104へ移行し、ステップ104以降において、第1の文字列S1及び第2の文字列S2を対応する一組の文字列とみなし、第1の文字列S1及び第2の文字列S2から単一の接尾辞木（構造接尾辞木と称する）を生成する処理を行う。なお、以下で生成される構造接尾辞木には、各枝に、第1の文字列S1の部分文字列に相当するラベル及び第2の文字列S2に相当するラベルが各々付される。

【0079】

また、以下において、 $\text{label}(V)$ はノードVからルートノードVrootに至る経路上の全ての枝に付されているラベルを順に結合したラベル（文字列）を表し、 $V(\cdots)$ は文字列「 \cdots 」をラベルとして持つ（ $\text{label}(V) = \text{「}\cdots\text{」}$ となる）ノードを表す。なお、 $V(\phi)$ はルートノードVrootを指す。また $E(\cdots)$ は、文字列「 \cdots 」が $\text{label}(V)$ の接頭辞となっているノードVと、ノードVの親ノードと、の間の間の枝を表し、 $f(\cdots)$ は、文字列「 \cdots 」のk番目（kは任意の整数）の文字（数値）がk以上の値のときにはk番目の文字（数値）を「0」に置き換える関数である。

【0080】

まずステップ104ではカウンタi及びカウンタjに「1」を各々代入し、ステップ106では構造接尾辞木のルートノードVrootを作成する。ステップ108では、構造接尾辞木を辿って検索を行う際の検索開始位置に相当するノードを指し示すためのパラメータsにVroot（ルートノード）を代入し、情報を一時的に記憶するためのテンポラリーtに ϕ （空集合）を代入し、特定のノードを指し

示すためのエリア $sL(Vroot)$ に $Vroot$ (ルートノード) を代入する。

【0081】

エリア $sL()$ は構造接尾辞木のうち、葉ノードを除く全てのノードに対応して各々設けられるエリアであり、ノード V のエリア $sL(V)$ には、 $label(W)$ が $label(V)$ から最初の 1 文字を除いた文字列となるノード W を指し示すための情報が設定される (すなわち $sL(V)=W$) か、又はノード W が存在しそうな枝の親ノードを指し示すための情報が設定される (詳細は後述) が、 $label(Vroot)=\phi$ であるため便宜的に $sL(Vroot)\leftarrow Vroot$ としている。

【0082】

次のステップ 110 ではカウンタ i の値が処理対象の文字列 S の文字数 n よりも大きいかな否かを判定する。判定が否定された場合にはステップ 112 で構造接尾辞木作成処理を行う (詳細は後述)。構造接尾辞木作成処理を行うと、ステップ 114 ではカウンタ i の値を「1」だけインクリメントしてステップ 110 に戻る。従って、 $i > n$ となる迄の間、カウンタ i の値をインクリメントしながら構造接尾辞木作成処理が繰り返し実行されることになる。

【0083】

次に、ステップ 112 の構造接尾辞木作成処理について、図 5 のフローチャートを参照して説明する。なお、理解が容易なように、処理対象の文字列 S が前出の (AUAUCGUUA\$) の場合 (すなわち、第 1 の文字列 $S1=(0022003\$)$ 、第 2 の文字列 $S2=(0111014\$)$ の場合) を例に説明するが、構造接尾辞木作成処理は任意の文字列の構造接尾辞木を作成可能であることは言うまでもない。

【0084】

ステップ 200 では、ノード s (最初はルートノード $Vroot$) から葉ノードの方向に構造接尾辞木を辿り、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝を探索する。例えば $i=j=1$ の場合には、 $S1[j..i-1]$ 及び $S2[j..i-1]$ は何れも ϕ であるので、 $f(S1[j..i-1])$ 及び $f(S2[j..i-1])$ も ϕ であると共に、このとき構造接尾辞木にはルートノード $Vroot$ 以外のノード及び枝が存在していないので、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝は発見されない。

【0085】

次のステップ 2 0 2 では、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝が有るか否か判定する。例えば $i = j = 1$ で $S1 = (0022003\$)$ 、 $S2 = (0111014\$)$ の場合、 $S1[j..i] = S1[1] = (0)$ 、 $S2[j..i] = S2[1] = (0)$ であるので、 $f(S1[j..i]) = (0)$ 、 $f(S2[j..i]) = (0)$ であるが、このとき構造接尾辞木にはルートノード V_{root} 以外のノード及び枝が存在していないので、前記判定は否定される。前記判定が否定された場合にはステップ 2 0 4 へ移行し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノードが有るか否か判定する。前述のように $i = j = 1$ であれば $f(S1[j..i-1])$ 、 $f(S2[j..i-1]) = \phi$ であるが、 $V(\phi)$ は V_{root} を指すので、この場合は前記判定が肯定されてステップ 2 0 6 へ移行し、フラグ `node#constructed` に「NO」を代入してステップ 2 1 2 へ移行する。

【0 0 8 6】

ステップ 2 1 2 ではテンポラリー t が ϕ か否か判定する。テンポラリー t はステップ 1 0 8 (図 2) で ϕ に初期設定されているので、この場合は前記判定が肯定されてステップ 2 1 8 へ移行し、フラグ `node#constructed` が「YES」か否か判定する。フラグ `node#constructed` は先のステップ 2 0 6 で「NO」が代入されているので、この場合は判定が否定されてステップ 2 2 0 へ移行し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード（この場合はルートノード V_{root} ）のエリア `sL()` が指しているノードをパラメータ s に代入する。`sL(V_{root})` には先のステップ 1 0 8 で V_{root} が代入されているので、パラメータ s にはルートノード V_{root} が代入される（次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される）。

【0 0 8 7】

ステップ 2 2 0 の処理を行うとステップ 2 2 6 へ移行し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード（この場合はルートノード V_{root} ）に子供のノードを作成し、作成した子供のノードとの間の枝に、 $f(S1[j..])$ から $f(S1[j..i-1])$ を除いたもの、及び $f(S2[j..])$ から $f(S2[j..i-1])$ を除いたものをラベルとして付与する。これにより、一例として図 6 (A) に示すように、ルートノードに子供のノード（葉ノード）が作成され、ルートノードと子供のノードとの間の枝（符号 A を付した枝 A）に、 $f(S1[j..])$ から $f(S1[j..i-1])$ を除いたもの

($f(S1[j..i-1]) = \phi$ であるので(0022003\$))、及び $f(S2[j..i-1])$ を除いたもの ($f(S2[j..i-1]) = \phi$ であるので(0111014\$))がラベルとして付与される。

【0088】

ステップ228ではカウンタjの値を「1」だけインクリメントし、次のステップ230ではカウンタjの値がカウンタiの値よりも大きいか否か判定する。このとき $i (= 1) < j (= 2)$ であるので、判定が肯定されて構造接尾辞木作成処理を一旦終了し、カウンタiが「1」だけインクリメント (図2のステップ114) された後に、 $i = j = 2$ の条件で構造接尾辞木作成処理が再度実行される。

【0089】

$i = j = 2$ の場合、 $S1[j..i-1]$, $S2[j..i-1] = \phi$ であるので、 $f(S1[j..i-1])$, $f(S2[j..i-1]) = \phi$ 、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝 = ϕ であるが、 $f(S1[j..i]) = f(S1[2]) = (0)$ 、 $f(S2[j..i]) = f(S2[2]) = (0)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝は存在する (枝A)。このため、ステップ202の判定が肯定されてステップ232へ移行し、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝の親ノード (この場合はルートノードVroot) をパラメータsに代入する。次のステップ234ではテンポラリtが ϕ か否か判定する。この場合は前記判定が肯定されて構造接尾辞木作成処理を一旦終了し、カウンタiがインクリメントされた後に、 $i = 3$, $j = 2$ の条件で構造接尾辞木作成処理が再度実行される。

【0090】

$i = 3$, $j = 2$ の場合、 $f(S1[j..i-1]) = f(S1[2]) = (0)$, $f(S2[j..i-1]) = f(S2[2]) = (0)$ であるので、ステップ200で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される (枝A)。また、 $f(S1[j..i]) = f(S1[2..3]) = (0)$ 、 $f(S2[j..i]) = f(S2[2..3]) = (01)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝も存在する (枝A)。このため、ステップ202の判定が再度肯定されてステップ232へ移行し、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝の親ノード (ルートノードVroot) をパラメータsに代入する (次にステップ

2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される)。また、テンポラリ t は ϕ であるので、ステップ 2 3 4 の判定が肯定されて構造接尾辞木作成処理を一旦終了し、カウンタ i がインクリメントされた後に、 $i = 4$ 、 $j = 2$ の条件で構造接尾辞木作成処理が再度実行される。

【0 0 9 1】

$i = 4$ 、 $j = 2$ の場合、 $f(S1[j..i-1]) = f(S1[2..3]) = (00)$ 、 $f(S2[j..i-1]) = f(S2[2..3]) = (01)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される (枝 A)。また、 $f(S1[j..i]) = f(S1[2..4]) = (002)$ 、 $f(S2[j..i]) = f(S2[2..4]) = (011)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝も存在する (枝 A)。このため、ステップ 2 0 2 の判定が再度肯定されてステップ 2 3 2 へ移行し、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝の親ノード (ルートノード V_{root}) をパラメータ s に代入する (次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される)。また、テンポラリ t は ϕ であるので、ステップ 2 3 4 の判定が肯定されて構造接尾辞木作成処理を一旦終了し、カウンタ i がインクリメントされた後に、 $i = 5$ 、 $j = 2$ の条件で構造接尾辞木作成処理が再度実行される。

【0 0 9 2】

$i = 5$ 、 $j = 2$ の場合、 $f(S1[j..i-1]) = f(S1[2..4]) = (002)$ 、 $f(S2[j..i-1]) = f(S2[2..4]) = (011)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される (枝 A)。また、 $f(S1[j..i]) = f(S1[2..5]) = (0020)$ 、 $f(S2[j..i]) = f(S2[2..5]) = (0110)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝 = ϕ 、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード = ϕ であり、ステップ 2 0 2 及びステップ 2 0 4 の判定が各々否定されてステップ 2 0 8 へ移行する。

【0 0 9 3】

ステップ 2 0 8 では、ステップ 2 0 0 における探索結果に基づき、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝 (この場合は枝 A) を分割し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ に相当するノードを作成する。これにより、枝

Aは、一例として図6 (B) に示すように、 $f(S1[j..i-1]) = (002)$ 及び $f(S2[j..i-1]) = (011)$ のラベルを付した枝 (枝A 1) と、元のラベルから $f(S1[j..i-1])$ 及び $f(S2[j..i-1])$ を除いたラベル (すなわち (2003\$) 及び (1014\$)) を付した枝 (枝A 2) と、に分割される。また、次のステップ 2 1 0 ではフラグ node#constructed に「YES」を代入してステップ 2 1 2 へ移行する。

【0 0 9 4】

このとき、テンポラリー t は ϕ であるのでステップ 2 1 2 の判定が肯定されてステップ 2 1 8 へ移行し、フラグ node#constructed は「YES」であるのでステップ 2 1 8 の判定が肯定されてステップ 2 2 2 へ移行する。ステップ 2 2 2 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード (この場合は枝A 1 と枝A 2 の間のノード) をテンポラリー t に記憶する。また、次のステップ 2 2 4 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノードの親ノード (この場合はルートノード Vroot) のエリア sL に記憶されているノードをパラメータ s に代入する (次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される)。

【0 0 9 5】

そして、次のステップ 2 2 6 において、枝A 1 と枝A 2 の間のノードに子供のノード (葉ノード) が作成され、枝A 1 と枝A 2 の間のノードと子供のノードとの間の枝 (枝B) に、 $f(S1[j..]) = f(S1[2..]) = (002003\$)$ から $f(S1[j..i-1]) = f(S1[2..4]) = (002)$ を除いたもの ($= (003\$)$)、及び $f(S2[j..]) = f(S2[2..]) = (011014\$)$ から $f(S2[j..i-1]) = f(S2[2..4]) = (011)$ を除いたもの ($= (014\$)$) がラベルとして付与される。

【0 0 9 6】

次のステップ 2 2 8 でカウンタ j をインクリメントすると $i = 5$, $j = 3$ となるので、ステップ 2 3 0 の判定が否定されてステップ 2 0 0 に戻り、 $i = 5$, $j = 3$ の条件で構造接尾辞木作成処理が再度実行される。

【0 0 9 7】

$i = 5$, $j = 3$ の場合、 $f(S1[j..i-1]) = f(S1[3..4]) = (00)$, $f(S2[j..i-1]) = f(S2[3..4]) = (01)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ

$E(f(S2[j..i-1]))$ の枝が発見される(枝A 1)。また、 $f(S1[j..i])=f(S1[3..5])=(000)$ 、 $f(S2[j..i])=f(S2[3..5])=(010)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝 $=\phi$ 、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード $=\phi$ であり、ステップ2 0 2 及びステップ2 0 4 の判定が各々否定されてステップ2 0 8 へ移行する。

【0 0 9 8】

ステップ2 0 8 では、ステップ2 0 0 における探索結果に基づき、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝(この場合は枝A 1)を分割し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ に相当するノードを作成する。これにより、枝A 1 は、一例として図6 (C) に示すように、 $f(S1[j..i-1])=(00)$ 及び $f(S2[j..i-1])=(01)$ のラベルを付した枝(枝A 3)と、元のラベルから $f(S1[j..i-1])$ 及び $f(S2[j..i-1])$ を除いたラベル(すなわち(2)及び(1))を付した枝(枝A 4)と、に分割される。また、次のステップ2 1 0 ではフラグnode#constructedに「YES」を代入してステップ2 1 2 へ移行する。

【0 0 9 9】

このとき、テンポラリーtには、先に実行したステップ2 2 2 により、旧枝A 1 (現在の枝A 4)と枝A 2 の間のノードが記憶されているので、ステップ2 1 2 の判定が否定されてステップ2 1 4 へ移行し、テンポラリーtに記憶されているノードのエリアsLに、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード(この場合は枝A 3 と枝A 4 との間のノード)を記憶する。次のステップ2 1 6 でテンポラリーtに ϕ を代入してステップ2 1 8 へ移行し、フラグnode#constructedは「YES」であるのでステップ2 1 8 の判定が肯定されてステップ2 2 2 へ移行する。

【0 1 0 0】

ステップ2 2 2 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード(この場合は枝A 3 と枝A 4 の間のノード)をテンポラリーtに記憶する。また、次のステップ2 2 4 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノードの親ノード(この場合はルートノードVroot)のエリアsLに記憶されているノードをパラメータsに代入する(次にステップ2 0 0 を実行するときには、このパ

ラメータ s に記憶されているノードから探索が開始される)。

【0 1 0 1】

そして、次のステップ 2 2 6 において、枝 A 3 と枝 A 4 の間のノードに子供のノード (葉ノード) が作成され、枝 A 3 と枝 A 4 の間のノードと子供のノードとの間の枝 (枝 C) に、 $f(S1[j..]) = f(S1[3..]) = (00003\$)$ から $f(S1[j..i-1]) = f(S1[3..4]) = (00)$ を除いたもの ($= (003\$)$)、及び $f(S2[j..]) = f(S2[3..]) = (00014\$)$ から $f(S2[j..i-1]) = f(S2[3..4]) = (01)$ を除いたもの ($= (014\$)$) がラベルとして付与される。

【0 1 0 2】

次のステップ 2 2 8 でカウンタ j をインクリメントすると $i = 5$, $j = 4$ となるので、ステップ 2 3 0 の判定が否定されてステップ 2 0 0 に戻り、 $i = 5$, $j = 4$ の条件で構造接尾辞木作成処理が再度実行される。

【0 1 0 3】

$i = 5$, $j = 4$ の場合、 $f(S1[j..i-1]) = f(S1[4]) = (0)$, $f(S2[j..i-1]) = f(S2[4]) = (0)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される (枝 A 3)。また、 $f(S1[j..i]) = f(S1[4..5]) = (00)$, $f(S2[j..i]) = f(S2[4..5]) = (00)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝 $= \phi$, $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード $= \phi$ であり、ステップ 2 0 2 及びステップ 2 0 4 の判定が各々否定されてステップ 2 0 8 へ移行する。

【0 1 0 4】

ステップ 2 0 8 では、ステップ 2 0 0 における探索結果に基づき、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝 (この場合は枝 A 3) を分割し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ に相当するノードを作成する。これにより、枝 A 3 は、一例として図 6 (D) に示すように、 $f(S1[j..i-1]) = (0)$ 及び $f(S2[j..i-1]) = (0)$ のラベルを付した枝 (枝 A 5) と、元のラベルから $f(S1[j..i-1])$ 及び $f(S2[j..i-1])$ を除いたラベル (すなわち (0) 及び (1)) を付した枝 (枝 A 6) と、に分割される。また、次のステップ 2 1 0 ではフラグ `node#constructed` に「YES」を代入してステップ 2 1 2 へ移行する。

【0 1 0 5】

このとき、テンポラリー t には、先に実行したステップ 2 2 2 により、旧枝 A 3（現在の枝 A 6）と枝 A 4 の間のノードが記憶されているので、ステップ 2 1 2 の判定が否定されてステップ 2 1 4 へ移行し、テンポラリー t に記憶されているノードのエリア sL に、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード（この場合は枝 A 5 と枝 A 6 との間のノード）を記憶する。次のステップ 2 1 6 でテンポラリー t に ϕ を代入してステップ 2 1 8 へ移行し、フラグ `node#constructed` は「YES」であるのでステップ 2 1 8 の判定が肯定されてステップ 2 2 2 へ移行する。

【0 1 0 6】

ステップ 2 2 2 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード（この場合は枝 A 5 と枝 A 6 の間のノード）をテンポラリー t に記憶する。また、次のステップ 2 2 4 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノードの親ノード（この場合はルートノード $Vroot$ ）のエリア sL に記憶されているノードをパラメータ s に代入する（次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される）。

【0 1 0 7】

そして、次のステップ 2 2 6 において、枝 A 5 と枝 A 6 の間のノードに子供のノード（葉ノード）が作成され、枝 A 5 と枝 A 6 の間のノードと子供のノードとの間の枝（枝 D）に、 $f(S1[j..]) = f(S1[4..]) = (0003\$)$ から $f(S1[j..i-1]) = f(S1[4]) = (0)$ を除いたもの（ $= (003\$)$ ）、及び $f(S2[j..]) = f(S2[4..]) = (0010\$)$ から $f(S2[j..i-1]) = f(S2[4]) = (0)$ を除いたもの（ $= (010\$)$ ）がラベルとして付与される。

【0 1 0 8】

次のステップ 2 2 8 でカウンタ j をインクリメントすると $i = 5$ ， $j = 5$ となるので、ステップ 2 3 0 の判定が否定されてステップ 2 0 0 に戻り、 $i = 5$ ， $j = 5$ の条件で構造接尾辞木作成処理が再度実行される。

【0 1 0 9】

$i = j = 5$ の場合、 $S1[j..i-1]$ ， $S2[j..i-1] = \phi$ 、 $E(f(S1[j..i-1]))$ かつ

$E(f(S2[j..i-1]))$ の枝 = ϕ であるが、 $S1[j..i] = S1[5] = (0)$ 、 $S2[j..i] = S2[5] = (0)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝は存在する（枝 A 5）。このため、ステップ 2 0 2 の判定が肯定されてステップ 2 3 2 へ移行し、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝の親ノード（この場合はルートノード Vroot）をパラメータ s に代入する（次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される）。

【0 1 1 0】

次のステップ 2 3 4 ではテンポラリ t が ϕ か否か判定する。このとき、テンポラリ t には、先に実行したステップ 2 2 2 により、枝 A 5 と枝 A 6 の間のノードが記憶されているので、ステップ 2 3 4 の判定が否定されてステップ 2 3 6 へ移行し、テンポラリ t に記憶されているノードのエリア sL に、パラメータ s に代入されているノード（この場合はルートノード Vroot）を記憶する。次のステップ 2 3 8 でテンポラリ t に ϕ を代入すると構造接尾辞木作成処理を一旦終了し、カウンタ i がインクリメントされた後に、 $i = 6$ 、 $j = 5$ の条件で構造接尾辞木作成処理が再度実行される。

【0 1 1 1】

$i = 6$ 、 $j = 5$ の場合、 $f(S1[j..i-1]) = f(S1[5]) = (0)$ 、 $f(S2[j..i-1]) = f(S2[5]) = (0)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される（枝 A 5）。また、 $S1[j..i] = S1[5..6] = (00)$ 、 $S2[j..i] = S2[5..6] = (01)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝も存在する（枝 A 6）。このため、ステップ 2 0 2 の判定が肯定されてステップ 2 3 2 へ移行し、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝の親ノード（この場合は枝 A 5 と枝 A 6 の間のノード）をパラメータ s に代入する（次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される）。

【0 1 1 2】

また、テンポラリ t には ϕ が代入されているので、ステップ 2 3 4 の判定が肯定されて構造接尾辞木作成処理を一旦終了し、カウンタ i がインクリメントされた後に、 $i = 7$ 、 $j = 5$ の条件で構造接尾辞木作成処理が再度実行される。

【 0 1 1 3 】

$i = 7$, $j = 5$ の場合、 $f(S1[j..i-1]) = f(S1[5..6]) = (00)$, $f(S2[j..i-1]) = f(S2[5..6]) = (01)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される（枝 A 6）。また、 $S1[j..i] = S1[5..7] = (000)$ 、 $S2[j..i] = S2[5..7] = (010)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝も存在する（枝 C）。このため、ステップ 2 0 2 の判定が肯定されてステップ 2 3 2 へ移行し、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝の親ノード（この場合は枝 A 6 と枝 C の間のノード）をパラメータ s に代入する（次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される）。

【 0 1 1 4 】

また、テンポラリ t には ϕ が代入されているので、ステップ 2 3 4 の判定が肯定されて構造接尾辞木作成処理を一旦終了し、カウンタ i がインクリメントされた後に、 $i = 8$, $j = 5$ の条件で構造接尾辞木作成処理が再度実行される。

【 0 1 1 5 】

$i = 8$, $j = 5$ の場合、 $f(S1[j..i-1]) = f(S1[5..7]) = (000)$, $f(S2[j..i-1]) = f(S2[5..7]) = (010)$ であるので、ステップ 2 0 0 で $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝が発見される（枝 C）。また、 $f(S1[j..i]) = f(S1[5..8]) = (000\$)$ 、 $f(S2[j..i]) = f(S2[5..8]) = (010\$)$ であるので、 $E(f(S1[j..i]))$ かつ $E(f(S2[j..i]))$ の枝 = ϕ 、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード = ϕ であり、ステップ 2 0 2 及びステップ 2 0 4 の判定が各々否定されてステップ 2 0 8 へ移行する。

【 0 1 1 6 】

ステップ 2 0 8 では、ステップ 2 0 0 における探索結果に基づき、 $E(f(S1[j..i-1]))$ かつ $E(f(S2[j..i-1]))$ の枝（この場合は枝 C）を分割し、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ に相当するノードを作成する。これにより、枝 C は、一例として図 6（E）に示すように、 $f(S1[j..i-1]) = (0)$ 及び $f(S2[j..i-1]) = (0)$ のラベルを付した枝（枝 C 1）と、元のラベルから $f(S1[j..i-1])$ 及び $f(S2[j..i-1])$ を除いたラベル（すなわち $(03\$)$ 及び $(14\$)$ ）を付した枝（枝 C

2) と、に分割される。また、次のステップ 2 1 0 ではフラグ node#constructed に「YES」を代入してステップ 2 1 2 へ移行する。

【0 1 1 7】

このとき、テンポラリー t は ϕ であるのでステップ 2 1 2 の判定が肯定されてステップ 2 1 8 へ移行し、フラグ node#constructed は「YES」であるのでステップ 2 1 8 の判定が肯定されてステップ 2 2 2 へ移行する。ステップ 2 2 2 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノード（この場合は枝 C 1 と枝 C 2 の間のノード）をテンポラリー t に記憶する。また、次のステップ 2 2 4 では、 $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノードの親ノード（この場合は枝 A 6 と枝 C 1 の間のノード）のエリア sL に記憶されているノードをパラメータ s に代入する（次にステップ 2 0 0 を実行するときには、このパラメータ s に記憶されているノードから探索が開始される）。

【0 1 1 8】

そして、次のステップ 2 2 6 において、枝 C 1 と枝 C 2 の間のノードに子供のノード（葉ノード）が作成され、枝 C 1 と枝 C 2 の間のノードと子供のノードとの間の枝（枝 E）に、 $f(S1[j..]) = f(S1[5..]) = (000\$)$ から $f(S1[j..i-1]) = f(S1[5..7]) = (000)$ を除いたもの（ $=(\$)$ ）、及び $f(S2[j..]) = f(S2[5..]) = (010\$)$ から $f(S2[j..i-1]) = f(S2[5..7]) = (010)$ を除いたもの（ $=(\$)$ ）がラベルとして付与される。上記により、 $j = 1 \sim 5$ の範囲について、第 1 の文字列 S1 及び第 2 の文字列 S2 の接尾辞 $f(S1[j..])$ 及び $f(S2[j..])$ が構造接尾辞木に組み入れられたことになる。

【0 1 1 9】

$j \geq 6$ の範囲については簡単に説明するが、接尾辞 $f(S1[6..]) = (00\$)$ 及び $f(S2[6..]) = (00\$)$ については、図 6 (F) に示すように枝 D が、 $f(S1[j..i-1]) = f(S1[6]) = (0)$ 及び $f(S2[j..i-1]) = f(S2[6]) = (0)$ のラベルを付した枝（枝 D 1）と、元のラベルから $f(S1[j..i-1])$ 及び $f(S2[j..i-1])$ を除いたラベル（すなわち $(03\$)$ 及び $(10\$)$ ）を付した枝（枝 D 2）に分割され、枝 D 1 と枝 D 2 の間のノードに子供のノードが作成され、両者のノードの間の枝 F に、 $f(S1[j..]) = f(S1[6..]) = (00\$)$ から $f(S1[j..i-1]) = f(S1[6..7]) = (00)$ を除いたもの

(=(\$))、及び $f(S2[j..])=f(S2[6..])=(000\$)$ から $f(S2[j..i-1])=f(S2[6..7])=(00)$ を除いたもの(=(\$))がラベルとして付与されることで構造接尾辞木に組み入れられる。

【0 1 2 0】

また、接尾辞 $f(S1[7..])=(0\$)$ 及び $f(S2[7..])=(0\$)$ については、図 6 (G) に示すように、枝 A 5 と枝 A 6 の間のノードに子供のノードが作成され、両者のノードの間の枝 G に、 $f(S1[j..])=f(S1[7..])=(0\$)$ から $f(S1[j..i-1])=f(S1[7])=(0)$ を除いたもの(=(\$))、及び $f(S2[j..])=f(S2[7..])=(0\$)$ から $f(S2[j..i-1])=f(S2[7])=(0)$ を除いたもの(=(\$))がラベルとして付与されることで構造接尾辞木に組み入れられる。

【0 1 2 1】

更に、接尾辞 $f(S1[8])=(\$)$ 及び $f(S2[8..])=(\$)$ については、図 6 (H) に示すように、ルートノードに子供のノードが作成され、両者のノードの間の枝 H に、 $f(S1[j..])=f(S1[8])=(\$)$ ($f(S1[j..i-1])=\phi$)、及び $f(S2[j..])=f(S2[8])=(\$)$ ($f(S2[j..i-1])=\phi$) がラベルとして付与されることで構造接尾辞木に組み入れられる。

【0 1 2 2】

なお、上記では構造接尾辞木作成処理の内容が容易に理解されるように、実際の処理に用いる処理対象の文字列と比較して長さが非常に短い文字列 S を用いて説明した関係上、エリア sL やテンポラリ t が有効に機能しているとは言い難いが、処理対象の文字列の長さが長くなり、生成される構造接尾辞木の規模が大きくなるに伴ってエリア sL やテンポラリ t が有効に機能し、エリア sL やテンポラリ t に記憶された情報に基づいてステップ 2 0 0 における探索がより低階層のノードから開始されることで、ノードの探索時間及び構造接尾辞木の作成時間が大幅に短縮されることになる。

【0 1 2 3】

上記のようにして処理対象の文字列 S の構造接尾辞木が生成されると、カウンタ i の値が n より大きくなることになってステップ 1 1 0 の判定が肯定され、構造接尾辞木生成処理を終了する。

【 0 1 2 4 】

続いて、処理対象の文字列 S の構造接尾辞木を利用した、文字列 S の構造の解析（文字列 S が塩基配列を表している場合、文字列 S の構造を解析することは前記塩基配列の構造を解析することと等価である）について説明する。なお、以下で説明する処理についても、コンピュータシステム 1 0 にインストールされた配列の構造解析プログラムによって実現される。

【 0 1 2 5 】

一本鎖 DNA や RNA の塩基配列の中に存在している構造が同一の部分配列は似た機能を持っている可能性が高く、配列自体が同一のもの及び配列自体は異なっているものを含めて構造が同一の部分配列が頻出していた場合、該部分配列は該部分配列は重要な機能を誘発する立体構造を示している可能性がある。このため、一本鎖 DNA や RNA の塩基配列中に頻出する部分配列を、配列自体が同一のもの及び配列自体は異なっているものを含め、構造が同一のものを同一の部分配列と見なして抽出することは、遺伝情報の解析に極めて重要な意味を持っている。これに対し、処理対象の文字列 S 中に頻出する部分文字列を、文字自体は異なっているものの構造が同一のものを含めて全て抽出することは、構造接尾辞木を利用すれば以下のような簡易な処理によって実現できる。

【 0 1 2 6 】

すなわち、構造接尾辞木において、或るノード V がその子孫（ノード V の子孫はノード V から見て葉の方向に存在する全てのノードを指す）に i 個の葉ノードを持つ場合、 $\text{label}(V)$ という部分文字列は処理対象の文字列 S の中に i 個存在する（接尾辞木も同様）。従って、例えば長さが m 以上で i 回以上頻出するような部分文字列を抽出することは $\text{label}(V)$ の長さが m 以上で、 i 個以上の葉ノードを子孫に持つようなノードを構造接尾辞木から全て検索することで実現できる。

【 0 1 2 7 】

なお、上記の検索方法は、接尾辞木から頻出する部分文字列を抽出する場合に利用される検索方法と同一であるが、本発明に係る構造接尾辞木を用いて上記の検索を行うことにより、文字自体は異なっているものの構造が同一のものを含め

、全ての部分文字列を抽出することができる。

【0 1 2 8】

また、遺伝情報の解析においては、二つの塩基配列中に各々存在する共通部分配列を、配列自体が同一か否かに拘わらず配列の構造が同一の部分配列を同一の部分配列と見なして抽出することも、先に説明した頻出する部分配列の抽出と同様に極めて重要な意味を持っている。これに対し、一対の文字列中に各々存在する共通部分文字列を、文字自体は異なっているものの構造が同一のものを含めて全て抽出することも、構造接尾辞木を利用すれば以下のような簡易な処理によって実現できる。

【0 1 2 9】

すなわち、まず共通部分文字列抽出対象の一対の文字列 S_1 , S_2 (第 1 の処理対象の配列及び第 2 の処理対象の配列) を結合し、以下のような文字列 S を作成する。

$$S = S_1 + '$ 1' + S_2 + '$ 2'$$

但し、 $+$ は結合を表し、 $' 1'$ は第 1 の末尾識別文字 (第 1 の識別情報)、 $' 2'$ は第 2 の末尾識別文字 (第 2 の識別情報) である。次に、上記の文字列 S を処理対象の文字列 S として前述の構造接尾辞木生成処理 ($\text{prev}(S)$ 及び $\text{compl}(S)$) の演算、構造接尾辞木の作成) を行う。

【0 1 3 0】

上記のようにして作成した構造接尾辞木において、葉ノード以外の或るノード V が、その子孫に、 $\text{label}(V_1)$ が $' 1'$ を含む葉ノード V_1 と、 $\text{label}(V_2)$ が $' 1'$ を含まない葉ノード V_2 と、を持つ場合、 $\text{label}(V)$ は文字列 S_1 , S_2 の共通部分文字列であると判断できる。従って、上記の条件に合致するノードを構造接尾辞木から検索して全て抽出することで、一対の文字列 S_1 , S_2 の共通部分文字列を、文字自体は異なっているものの構造が同一のものを含めて全て抽出することができる。

【0 1 3 1】

なお、上記では処理対象の文字列 S に対して $\text{prev}(S)$ 及び $\text{compl}(S)$ を演算することで求めた第 1 の文字列 S_1 及び第 2 の文字列 S_2 を対応する一組の文字列と

みなし、各枝に第 1 の文字列 S_1 の部分文字列に相当するラベル及び第 2 の文字列 S_2 に相当するラベルを各々付した構造接尾辞木を作成していたが、これに限定されるものではない。、

【0 1 3 2】

すなわち、 $\text{prev}(S)$ 及び $\text{compl}(S)$ を演算することで求めた第 1 の文字列 S_1 及び第 2 の文字列 S_2 の一方（文字列 A とする）に存在する全ての「0」を、他方の文字列（文字列 B とする）において前記「0」と同一の位置に存在する数値の正負の符号を反転した値に置き換えるか、又は処理対象の文字列を前記置き換えによって得られる文字列と同一の文字列へ直接変換するアルゴリズムに従って処理対象の文字列を変換し、得られた単一の文字列を用いて構造接尾辞を作成するようにしてもよい。この場合、各枝に付すラベルとして付す文字列の情報量を少なくすることができ、構造接尾辞木を記憶するための記憶容量を節減することができる。

【0 1 3 3】

また、図 5 に示した構造接尾辞木作成処理では、葉ノード以外のノードを新たに作成（ステップ 2 0 8）する毎（フラグ node\#constructed に「YES」を代入する毎）に、 $V(f(S_1[j..i-1]))$ かつ $V(f(S_2[j..i-1]))$ のノードをテンポラリ t に記憶し（ステップ 2 2 2）、次に構造接尾辞木作成処理を実行する際に、テンポラリ t に記憶されているノードのエリア sL に、 $V(f(S_1[j..i-1]))$ かつ $V(f(S_2[j..i-1]))$ のノード（ステップ 2 1 4）、又はパラメータ s に代入されているノード（ステップ 2 3 6）を記憶し、ステップ 2 0 0 における枝の探索における探索開始ノードの決定に用いるようにしていたが、これに限定されるものではなく、構造接尾辞木の各枝毎にノードの集合 T （初期値は ϕ ）を設定し、この集合 T を用いて以下のようにエリア sL にノードを記憶するようにしてもよい。

【0 1 3 4】

すなわち、葉ノード以外のノードを新たに作成（ステップ 2 0 8）する毎に、 $E(f(S_1[j..i-1]))$ かつ $E(f(S_2[j..i-1]))$ の枝（枝 e_0 とする）に対応する集合 $T(e_0)$ に属する全てのノードを、そのラベルの長さに 1 を加えた値が「 $i-2$ 」より小さいか「 $i-1$ 」より大きいかに応じて分類し、ステップ 2 0 8 における枝

の分割によって生じた 2 つの枝のうち、ルートノード V_{root} に近い方の枝を枝 e_1 、ルートノード V_{root} から遠い方の枝を枝 e_2 としたときに、集合 $T(e_0)$ に属するノードのうち、ラベルの長さが短い（「 $i-2$ 」より小さい）と判断したノードを枝 e_1 の集合 $T(e_1)$ に割り当て、ラベルの長さが長い（「 $i-1$ 」より大きい）と判断したノードを枝 e_2 の集合 $T(e_2)$ に割り当てる。そして、集合 $T(e_2)$ に割り当てられた全てのノード v について、エリア $sL(v)$ に $V(f(S1[j..i-1]))$ かつ $V(f(S2[j..i-1]))$ のノードを記憶する。また、ステップ 2 3 6 において、テンポラリ t に記憶されているノードのエリア sL に、パラメータ s に代入されているノードを記憶した後に、テンポラリ t に記憶されているノードを集合 $T(E(f(S1[j..i-1]))$ かつ $E(f(S1[j..i-1]))$ に追加する。

【0 1 3 5】

上記により、構造接尾辞木作成処理自体は若干複雑になるものの、ステップ 2 0 0 における探索をより低階層のノードから開始することができ、ノードの探索時間及び構造接尾辞木の作成時間を更に短縮することができる。

【0 1 3 6】

また、上記では処理対象の配列の構成要素を文字に置き換えることで得られる処理対象の配列を表す文字列を用いて処理対象の配列の構造を解析する場合を説明したが、本発明を実施するにあたり、配列の構成要素は任意の情報に置き換え可能であり、処理対象の配列を表す任意の情報列を用いて処理対象の配列の構造を解析可能であることは言うまでもない。

【0 1 3 7】

更に、上記では DNA や RNA の塩基配列の構造の解析に本発明を適用した場合を例に説明したが、本発明は上記に限定されるものではなく、類似の構造を持つポリマーの配列の構造を解析する等の場合にも適用可能である。

【0 1 3 8】

また、上記では本発明に係る配列の変換方法及び配列の構造解析方法を実現するための配列の構造解析プログラムが、当初は、本発明に係る記録媒体としての情報記憶媒体 6 0 に記憶されており、情報記憶媒体 6 0 から本実施形態に係るコンピュータシステム 1 0 へ前記プログラムがインストールされて実行されること

により、コンピュータシステム 1 0 が本発明に係る配列の構造解析装置として機能する態様について説明したが、前記プログラムを、当初は公衆電話回線やコンピュータネットワーク（例えば LAN、インターネット、無線通信ネットワーク等）システムにおける通信媒体（光ファイバや無線回線）を介してコンピュータシステム 1 0 と接続される他の情報処理機器（例えばネットワークサーバ）の記憶装置に記憶しておき、コンピュータシステム 1 0 が前記情報処理機器と通信することで、前記情報処理機器から前記通信媒体（本発明に係る伝送媒体）によって前記プログラムがコンピュータシステム 1 0 へ伝送され、伝送されたプログラムをコンピュータシステム 1 0 が HDD 4 0 等の記憶手段にインストールして実行することにより、コンピュータシステム 1 0 が本発明に係る配列の構造解析装置として機能するようにしてもよい。

【 0 1 3 9 】

【実施例】

次に、本願発明者が実施した実験の結果について説明する。

【 0 1 4 0 】

〔第 1 実施例〕

【 0 1 4 1 】

H I V（ヒト免疫不全ウイルス）の RNA 全配列（配列のアクセッション番号：K03455，長さ：9 7 1 9）を以下に示す。

```

tggaagggt aattcactcc caacgaagac aagatatacct tgatctgttg atctaccaca
cacaaggcta cttccctgat tagcagaact acacaccagg gccagggatc agatatccac
tgacctttgg atggtgctac aagctagtag cagttgagcc agagaagtta gaagaagcca
acaaaggaga gaacaccagc ttgttacacc ctgtgagcct gcatggaatg gatgaccggg
agagagaagt gttagagtgg aggtttgaca gccgcctagc atttcatcac atggcccagag
agctgcatcc ggagtacttc aagaactgct gacatcgagc ttgctacaag ggactttccg
ctggggactt tccaggagg cgtggcctgg gcgggactgg ggagtggcga gccctcagat
cctgcatata agcagctgct ttttgctgt actgggtctc tctggttaga ccagatctga
gcctgggagc tctctggcta actagggaac ccactgctta agcctcaata aagcttgcct
tgagtgcctc aagtagtgtg tgcccgtctg ttgtgtgact ctggtaacta gagatccctc

```


agaccctttt agtcagtgtg gaaaatctct agcagtggcg cccgaacagg gacctgaaag
 cgaaagggaa accagaggag ctctctcgac gcaggactcg gcttgctgaa gcgcgcacgg
 caagaggcga ggggcggcga ctggtgagta cgccaaaaat ttgactagc ggaggctaga
 aggagagaga tgggtgcgag agcgtcagta ttaagcgggg gagaattaga tcgatgggaa
 aaaattcggg taaggccagg gggaaagaaa aaatataaat taaaacatat agtatgggca
 agcaggggagc tagaacgatt cgcagttaat cctggcctgt tagaaacatc agaaggctgt
 agacaaatac tgggacagct acaaccatcc cttcagacag gatcagaaga acttagatca
 ttatataata cagtagcaac cctctattgt gtgcatcaaa ggatagagat aaaagacacc
 aaggaagctt tagacaagat agaggaagag caaaacaaaa gtaagaaaaa agcacagcaa
 gcagcagctg acacaggaca cagcaatcag gtcagccaaa attaccctat agtgcagaac
 atccaggggc aaatggtaca tcaggccata tcacctagaa ctttaaatgc atgggtaaaa
 gtagtagaag agaaggcttt cagcccagaa gtgataccca tgttttcagc attatcagaa
 ggagccaccc cacaagattt aaacaccatg ctaaacacag tggggggaca tcaagcagcc
 atgcaaatgt taaaagagac catcaatgag gaagctgcag aatgggatag agtgcatcca
 gtgcatgcag ggcctattgc accaggccag atgagagaac caaggggaag tgacatagca
 ggaactacta gtacccttca ggaacaaata ggatggatga caaataatcc acctatccca
 gtaggagaaa tttataaaag atggataatc ctgggattaa ataaaatagt aagaatgtat
 agccctacca gcattctgga cataagacaa ggaccaaagg aaccctttag agactatgta
 gaccggttct ataaaactct aagagccgag caagcttcac aggaggtaaa aaattggatg
 acagaaacct tgttgggtcca aaatgcgaac ccagattgta agactatttt aaaagcattg
 ggaccagcgg ctacactaga agaaatgatg acagcatgtc agggagtagg aggaccggc
 cataaggcaa gaggtttggc tgaagcaatg agccaagtaa caaattcagc taccataatg
 atgcagagag gcaatttttag gaaccaaaga aagattgtta agtgtttcaa ttgtggcaaa
 gaagggcaca cagccagaaa ttgcagggcc cctaggaaaa agggctgttg gaaatgtgga
 aaggaaggac accaaatgaa agattgtact gagagacagg ctaatttttt agggaagatc
 tggccttcct acaagggaag gccagggaat tttcttcaga gcagaccaga gccaacagcc
 ccaccagaag agagcttcag gtctggggta gagacaacaa ctccccctca gaagcaggag
 ccgatagaca aggaactgta tcctttaact tccctcaggt cactcttttg caacgacccc
 tcgtcacaat aaagataggg gggcaactaa aggaagctct attagataga ggagcagatg

atacagtatt agaagaaatg agtttgccag gaagatggaa accaaaaatg ataggggggaa
 ttggagggttt tatcaaagta agacagtatg atcagatact catagaaatc tgtggacata
 aagctatagg tacagtatta gtaggaccta cacctgtcaa cataattgga agaaatctgt
 tgactcagat tggttgcact ttaaattttc ccattagccc tattgagact gtaccagtaa
 aattaaagcc aggaatggat ggcccaaaag ttaaacaatg gccattgaca gaagaaaaaa
 taaaagcatt agtagaaatt tgtacagaga tggaaaagga agggaaaatt tcaaaaattg
 ggcctgaaaa tccatacaat actccagtat ttgccataaa gaaaaaagac agtactaaat
 ggagaaaaatt agtagatttc agagaactta ataagagaac tcaagacttc tgggaagttc
 aattaggaat accacatccc gcagggttaa aaaagaaaaa atcagtaaca gtactggatg
 tgggtgatgc atatttttca gttcccttag atgaagactt caggaagtat actgcattta
 ccatacctag tataaacaat gagacaccag ggattagata tcagtacaat gtgcttcac
 agggatggaa aggatcacca gcaatatcc aaagtagcat gacaaaaatc ttagagcctt
 ttagaaaaaca aaatccagac atagttatct atcaatacat ggatgatttg tatgtaggat
 ctgacttaga aatagggcag catagaacaa aaatagagga gctgagacaa catctgttga
 ggtgggggact taccacacca gacaaaaaac atcagaaaga acctccattc ctttggatgg
 gttatgaact ccatcctgat aaatggacag tacagcctat agtgctgcca gaaaaagaca
 gctggactgt caatgacata cagaagttag tggggaaatt gaattgggca agtcagattt
 acccagggat taaagtaagg caattatgta aactccttag aggaaccaa gcactaacag
 aagtaatacc actaacagaa gaagcagagc tagaactggc agaaaacaga gagattctaa
 aagaaccagt acatggagtg tattatgacc catcaaaaga cttaatagca gaaatacaga
 agcaggggca aggccaatgg acatatcaaa tttatcaaga gccatttaaa aatctgaaaa
 caggaaaaata tgcaagaatg aggggtgccc acactaatga tgtaaaacaa ttaacagagg
 cagtgcacaaa aataaccaca gaaagcatag taatatgggg aaagactcct aaatttaaac
 tgcccataca aaaggaaaca tgggaaacat ggtggacaga gtattggcaa gccacctgga
 ttcttgagtg ggagtttggt aatacccctc ccttagtgaa attatggtac cagttagaga
 aagaacccat agtaggagca gaaaccttct atgtagatgg ggcagctaac agggagacta
 aattaggaag agcaggatat gttactaata gaggaagaca aaaagtgtgc accctaactg
 acacaacaaa tcagaagact gagttacaag caatttatct agctttgcag gattcgggat
 tagaagtaaa catagtaaca gactcacaat atgcattagg aatcattcaa gcacaaccag

atcaaagtga atcagagtta gtcaatcaaa taatagagca gttaataaaa aaggaaaagg
 tctatctggc atgggtacca gcacacaaag gaattggagg aaatgaacaa gtagataaat
 tagtcagtgc tggaatcagg aaagtactat ttttagatgg aatagataag gccaagatg
 aacatgagaa atatcacagt aattggagag caatggctag tgattttaac ctgccacctg
 tagtagcaaa agaaatagta gccagctgtg ataaatgtca gctaaaagga gaagccatgc
 atggacaagt agactgtagt ccaggaatat ggcaactaga ttgtacacat ttagaaggaa
 aagttatcct ggtagcagtt catgtagcca gtggatatat agaagcagaa gttattccag
 cagaaacagg gcaggaaaca gcatattttc ttttaaaatt agcaggaaga tggccagtaa
 aaacaataca tactgacaat ggcagcaatt tcaccggtgc tacggttagg gccgcctgtt
 ggtgggcggg aatcaagcag gaatttggaa ttccctacaa tcccaaagt caaggagtag
 tagaatctat gaataaagaa ttaaagaaaa ttataggaca ggtaagagat caggctgaac
 atcttaagac agcagtacaa atggcagtat tcatccacaa ttttaaaaga aaagggggga
 ttggggggta cagtgcaggg gaaagaatag tagacataat agcaacagac atacaaacta
 aagaattaca aaaacaaatt acaaaaattc aaaattttcg ggtttattac agggacagca
 gaaatccact ttggaaagga ccagcaaagc tcctctggaa aggtgaaggg gcagtagtaa
 tacaagataa tagtgacata aaagtagtgc caagaagaaa agcaaagatc attagggatt
 atggaaaaca gatggcaggt gatgattgtg tggcaagtag acaggatgag gattagaaca
 tggaaaagt tttagtaaaaca ccatatgtat gtttcaggga aagctagggg atggttttat
 agacatcact atgaaagccc tcatccaaga ataagttcag aagtacacat cccactaggg
 gatgctagat tggtaataac aacatattgg ggtctgcata caggagaaag agactggcat
 ttgggtcagg gagtctccat agaattggagg aaaaagagat atagcacaca agtagaccct
 gaactagcag accaactaat tcatctgtat tactttgact gtttttcaga ctctgctata
 agaaaggcct tattaggaca catagttagc cctaggtgtg aatatcaagc aggacataac
 aaggtaggat ctctacaata cttggcacta gcagcattaa taacaccaaa aaagataaag
 ccacctttgc ctagtgttac gaaactgaca gaggatagat ggaacaagcc ccagaagacc
 aagggccaca gagggagcca cacaatgaat ggacactaga gcttttagag gagcttaaga
 atgaagctgt tagacatttt cctaggattt ggctccatgg cttagggcaa catatctatg
 aaacttatgg ggatacttgg gcaggagtgg aagccataat aagaattctg caacaactgc
 tgtttatcca ttttcagaat tgggtgtcga catagcagaa taggcgttac tcgacagagg



agagcaagaa atggagccag tagatcctag actagagccc tggaagcatc caggaagtca
 gcctaaaact gcttgtacca attgctattg taaaaagtgt tgctttcatt gccaaagtttg
 tttcataaca aaagccttag gcatctccta tggcaggaag aagcggagac agcgacgaag
 agctcatcag aacagtcaga ctcatacaagc ttctctatca aagcagtaag tagtacatgt
 aacgcaacct ataccaatag tagcaatagt agcattagta gtagcaataa taatagcaat
 agttgtgtgg tccatagtaa tcatagaata taggaaaata ttaagacaaa gaaaaataga
 cagggttaatt gatagactaa tagaaagagc agaagacagt ggcaatgaga gtgaaggaga
 aatatcagca cttgtggaga tgggggtgga gatggggcac catgctcctt gggatgttga
 tgatctgtag tgctacagaa aaattgtggg tcacagtcta ttatggggta cctgtgtgga
 aggaagcaac caccactcta ttttgtgcat cagatgctaa agcatatgat acagaggtac
 ataatgtttg ggccacacat gcctgtgtac ccacagacc caaccacaa gaagtagtat
 tggtaaatgt gacagaaaat tttaacatgt ggaaaaatga catggtagaa cagatgcatg
 aggatataat cagtttatgg gatcaaagcc taaagccatg tgtaaaatta accccactct
 gtgttagttt aaagtgcact gatttgaaga atgatactaa taccaatagt agtagcgga
 gaatgataat ggagaaagga gagataaaaa actgctcttt caatatcagc acaagcataa
 gaggttaaggt gcagaaagaa tatgcatttt tttataaact tgatataata ccaatagata
 atgatactac cagctataag ttgacaagtt gtaacacctc agtcattaca caggcctgtc
 caaaggatc ctttgagcca attcccatac attattgtgc cccggctggt tttgcgattc
 taaaatgtaa taataagacg ttcaatggaa caggaccatg tacaaatgtc agcacagtac
 aatgtacaca tggaattagg ccagtagtat caactcaact gctgttaaat ggcagtctag
 cagaagaaga ggtagtaatt agatctgtca atttcacgga caatgctaaa accataatag
 tacagctgaa cacatctgta gaaattaatt gtacaagacc caacaacaat acaagaaaaa
 gaatccgtat ccagagagga ccaggagag catttggtac aataggaaaa ataggaaata
 tgagacaagc acattgtaac attagtagag caaaatggaa taacacttta aaacagatag
 ctagcaaat aagagaacaa tttggaaata ataaaacaat aatctttaag caatcctcag
 gaggggaccc agaaattgta acgcacagtt ttaattgtgg aggggaattt ttctactgta
 attcaacaca actgtttaat agtacttggt ttaatagtag ttggagtact gaagggtcaa
 ataacactga aggaagtac acaatcacc tcccatgcag aataaaacaa attataaaca
 tgtggcagaa agtaggaaaa gcaatgtatg cccctcccat cagtggacaa attagatgtt

catcaaatat tacagggttg ctattaacaa gagatgggtg taatagcaac aatgagtcg
 agatcttcag acctggagga ggagatatga gggacaattg gagaagtga ttatataaat
 ataaagtagt aaaaattgaa ccattaggag tagcaccac caaggcaaag agaagagtgg
 tgcagagaga aaaaagagca gtgggaatag gagctttgtt ccttgggttc ttgggagcag
 caggaagcac tatgggcgca gcctcaatga cgctgacggt acaggccaga caattattgt
 ctggtatagt gcagcagcag aacaatttgc tgagggtat tgaggcgcaa cagcatctgt
 tgcaactcac agtctggggc atcaagcagc tccaggcaag aatcctggct gtggaaagat
 acctaaagga tcaacagctc ctggggattt ggggttgctc tggaaaactc atttgcacca
 ctgctgtgcc ttggaatgct agttggagta ataaatctct ggaacagatt tggaatcaca
 cgacctggat ggagtgggac agagaaatta acaattacac aagcttaata cactccttaa
 ttgaagaatc gcaaaaccag caagaaaaga atgaacaaga attattggaa ttagataaat
 gggcaagttt gtggaattgg tttaacataa caaattggct gtggtatata aaattattca
 taatgatagt aggaggcttg gtaggtttta gaatagtttt tgctgtactt tctatagtga
 atagagttag gcagggatat tcaccattat cgtttcagac ccacctcca accccgaggg
 gacccgacag gcccgaggga atagaagaag aagggtggaga gagagacaga gacagatcca
 ttcgattagt gaacggatcc ttggcactta tctgggacga tctgcggagc ctgtgcctct
 tcagctacca ccgcttgaga gacttactct tgattgtaac gaggattgtg gaacttctgg
 gacgcagggg gtgggaagcc ctcaaattt ggtggaatct cctacagtat tggagtcagg
 aactaaagaa tagtgctgtt agcttgctca atgccacagc catagcagta gctgagggga
 cagatagggt tatagaagta gtacaaggag cttgtagagc tattcgccac atacctagaa
 gaataagaca gggcttggaaggat ttttgc tataagatgg gtggcaagt gtcaaaaagt
 agtgtgattg gatggcctac tgtaaggga agaattgagac gagctgagcc agcagcagat
 aggggtgggag cagcatctcg agacctgga aaacatggag caatcacaag tagcaataca
 gcagctacca atgctgcttg tgcctggcta gaagcacaag aggaggagga ggtgggtttt
 ccagtcacac ctcaggatcc tttaagacca atgacttaca aggcagctgt agatcttagc
 cactttttta aagaaaagg gggactggaa gggctaattc actcccaaag aagacaagat
 atccttgatc tgtggatcta ccacacacaa ggctacttcc ctgattagca gaactacaca
 ccagggccag gggtcagata tccactgacc tttggatggg gctacaagct agtaccagtt
 gagccagata agatagaaga ggccaataaa ggagagaaca ccagcttggt acaccctgtg

agcctgcatg ggatggatga cccggagaga gaagtgttag agtggaggtt tgacagccgc
ctagcatttc atcacgtggc ccgagagctg catccggagt acttcaagaa ctgctgacat
cgagcttgct acaagggact ticcgtggg gactttccag ggaggcgtgg cctgggcggg
actggggagt ggcgagccct cagatcctgc atataagcag ctgctttttg cctgtactgg
gtctctctgg ttagaccaga tctgagcctg ggagctctct ggctaactag ggaaccact
gcttaagcct caataaagct tgccttgagt gcttcaagta gtgtgtgccc gtctgttggtg
tgactctggt aactagagat cctcagacc cttttagtca gtgtggaaaa tctctagca

【 0 1 4 2 】

本願発明者は、上記の配列に対し、本発明を適用して長さ 10 以上でかつ出現回数が 3 回以上の全てのパターンを検索する実験を行った。検索結果を以下に示す。

=== 1st pattern: length = 10

55: CCACACACAA 9140: CCACACACAA 238: GGAGAGAGAA 9323: GGAGAGAGAA

=== 2nd pattern: length = 10

259: GAGGTTTGAC 9344: GAGGTTTGAC 345: ACAAGGGACT 9430: ACAAGGGACT

=== 3rd pattern: length = 10

349: GGGACTTTCC 9434: GGGACTTTCC 363: GGGACTTTCC 9448: GGGACTTTCC
5131: TTTCAGGGAA

=== 4th pattern: length = 10

294: CCCGAGAGCT 9379: CCCGAGAGCT 622: AAATCTCTAG 9707: AAATCTCTAG

=== 5th pattern: length = 10

485: GGAGCTCTCT 9570: GGAGCTCTCT 676: GGAGCTCTCT

=== 6th pattern: length = 10

227: ATGGATGACC 9312: ATGGATGACC 700: GCTTGCTGAA

=== 7th pattern: length = 10

54: ACCACACACA 9139: ACCACACACA 780: AGGAGAGAGA

=== 8th pattern: length = 10

26: AGACAAGATA 9111: AGACAAGATA 1091: AGACAAGATA

=== 9th pattern: length = 11

274: CCTAGCATTTTC 9359: CCTAGCATTTTC 1345: CCATGCTAAAC
 ===10th pattern: length = 10
 13: TCACTCCCAA 1739: GACAGAAACC 9098: TCACTCCCAA
 ===11th pattern: length = 10
 93: CACCAGGGCC 9178: CACCAGGGCC 1849: GAGGACCCGG
 ===12th pattern: length = 10
 1948: GAAAGATTGT 2057: GAAAGATTGT 5591: AGGGAGCCAC
 ===13th pattern: length = 10
 343: CTACAAGGGA 9428: CTACAAGGGA 2108: CTACAAGGGA
 ===14th pattern: length = 10
 567: CTGTTGTGTG 9652: CTGTTGTGTG 2164: CAGAAGAGAG
 ===15th pattern: length = 10
 943: AAACATCAGA 3206: AAACATCAGA 2277: CCCTCGTCAC 7369: TTTCTACTGT
 ===16th pattern: length = 11
 1817: AGAAGAAATGA 2350: AGAAGAAATGA 3962: ACAACAAATCA
 ===17th pattern: length = 10
 414: TCAGATCCTG 9499: TCAGATCCTG 2567: ACTGTACCAG
 ===18th pattern: length = 10
 515: GCTTAAGCCT 9600: GCTTAAGCCT 2617: ATGGCCATTG
 ===19th pattern: length = 10
 863: AAAGAAAAAA 2851: AAAGAAAAAA 2737: AAAGAAAAAA
 ===20th pattern: length = 10
 864: AAGAAAAAAT 2631: AAGAAAAAAT 2852: AAGAAAAAAT
 ===21st pattern: length = 10
 2484: GACCTACACC 2873: CTGGATGTGG 4066: TCAAGCACAA
 ===22nd pattern: length = 10
 335: CGAGCTTGCT 9420: CGAGCTTGCT 3095: TACATGGATG
 ===23rd pattern: length = 10
 2739: AGAAAAAAGA 3201: ACAAAAAACA 7747: AGAAAAAAGA

===24th pattern: length = 11

2276: CCCCTCGTCAC 3205: AAAACATCAGA 7368: TTTTCTACTGT

===25th pattern: length = 11

123: CCTTTGGATGG 9208: CCTTTGGATGG 3229: CCTTTGGATGG

===26th pattern: length = 11

353: CTTTCCGCTGG 9438: CTTTCCGCTGG 3237: TGGGTTATGAA

===27th pattern: length = 10

354: TTTCCGCTGG 9439: TTTCCGCTGG 3238: GGGTTATGAA

===28th pattern: length = 10

405: GGCGAGCCCT 9490: GGCGAGCCCT 3383: TTATGTAAAC

===29th pattern: length = 10

3326: TTAGTGGGGA 3457: GGCAGAAAAC 3616: AATGAGGGGT

===30th pattern: length = 10

554: AGTGTGTGCC 9639: AGTGTGTGCC 3466: CAGAGAGATT

===31st pattern: length = 10

395: ACTGGGGAGT 9480: ACTGGGGAGT 3593: CTGAAAACAG

===32nd pattern: length = 10

2646: CATTAGTAGA 3633: CTAATGATGT 7219: CATTAGTAGA

===33rd pattern: length = 10

611: GTCAGTGTGG 9696: GTCAGTGTGG 9001: CAGTCACACC 3956: ACTGACACAA

===34th pattern: length = 11

86: AACTACACACC 9171: AACTACACACC 4157: CCAGCACACAA

===35th pattern: length = 10

608: TTAGTCAGTG 9693: TTAGTCAGTG 4199: TTAGTCAGTG

===36th pattern: length = 10

83: CAGAACTACA 9168: CAGAACTACA 4383: GACAAGTAGA

===37th pattern: length = 10

65: GGCTACTTCC 9150: GGCTACTTCC 4538: TTAGCAGGAA

===38th pattern: length = 10



2895: TTTCAGTTCC 4665: AAAGTCAAGG 9080: GGGACTGGAA

===39th pattern: length = 10

4524: ATTTTCTTTT 4783: TAAAAGAAAA 9067: TAAAAGAAAA

===40th pattern: length = 10

585: AACTAGAGAT 9670: AACTAGAGAT 5200: AAGTACACAT

===41st pattern: length = 10

307: TCCGGAGTAC 9392: TCCGGAGTAC 5742: GAATTCTGCA

===42nd pattern: length = 11

369: TTCCAGGGAGG 9454: TTCCAGGGAGG 5931: CCAAGTTTGTT

===43rd pattern: length = 10

370: TCCAGGGAGG 9455: TCCAGGGAGG 5932: CAAGTTTGTT

===44th pattern: length = 10

3848: ATAGTAGGAG 5991: GCGACGAAGA 8285: ATAGTAGGAG

===45th pattern: length = 10

6078: AGTAGCAATA 6099: AGTAGCAATA 8928: AGTAGCAATA

===46th pattern: length = 10

90: ACACACCAGG 9175: ACACACCAGG 6123: TGTGTGGTCC 7152: AGAGAGGACC

===47th pattern: length = 10

3601: AGGAAAATAT 5151: TGGTTTTATA 6151: AGGAAAATAT

===48th pattern: length = 10

1105: AAGAGCAAAA 6255: GGAGATGGGG 6267: GGAGATGGGG

===49th pattern: length = 10

3874: AGATGGGGCA 6269: AGATGGGGCA 7227: GAGCAAAATG

===50th pattern: length = 10

6047: AAGTAGTACA 6296: TTGATGATCT 8715: AAGTAGTACA

===51st pattern: length = 13

42: ATCTGTGGATCTA 9127: ATCTGTGGATCTA 6409: TACAGAGGTACAT

===52nd pattern: length = 10

235: CCCGGAGAGA 9320: CCCGGAGAGA 6429: GGGCCACACA

===53rd pattern: length = 11
 589: AGAGATCCCTC 9674: AGAGATCCCTC 6443: TGTGTACCCAC
 ===54th pattern: length = 10
 590: GAGATCCCTC 9675: GAGATCCCTC 6444: GTGTACCCAC
 ===55th pattern: length = 10
 6294: TGTGATGAT 8713: AGAAGTAGTA 6469: AGAAGTAGTA
 ===56th pattern: length = 10
 470: CCAGATCTGA 9555: CCAGATCTGA 6611: AAGTGCACTG
 ===57th pattern: length = 10
 182: AAAGGAGAGA 9267: AAAGGAGAGA 6674: AAAGGAGAGA
 ===58th pattern: length = 10
 6070: ATACCAATAG 6639: ATACCAATAG 6767: ATACCAATAG
 ===59th pattern: length = 11
 418: ATCCTGCATAT 9503: ATCCTGCATAT 6787: TACCAGCTATA
 ===60th pattern: length = 10
 419: TCCTGCATAT 9504: TCCTGCATAT 6788: ACCAGCTATA
 ===61st pattern: length = 11
 121: GACCTTTGGAT 9206: GACCTTTGGAT 6837: GTCCAAAGGTA
 ===62nd pattern: length = 10
 587: CTAGAGATCC 9672: CTAGAGATCC 6963: GTACACATGG
 ===63rd pattern: length = 10
 153: TTGAGCCAGA 7052: TTCACGGACA 9238: TTGAGCCAGA
 ===64th pattern: length = 10
 10: AATTCACCTCC 9095: AATTCACCTCC 7843: GGCCAGACAA
 ===65th pattern: length = 10
 356: TCCGCTGGGG 9441: TCCGCTGGGG 8310: GAATAGTTTT
 ===66th pattern: length = 10
 6348: TACCTGTGTG 7148: ATCCAGAGAG 8532: GCTTGAGAGA
 ===67th pattern: length = 10

7965: TGGCTGTGGA 8542: CTTACTCTTG 8920: CAATCACAAG

===68th pattern: length = 10

115: TCCACTGACC 9200: TCCACTGACC 8630: TGGAGTCAGG

===69th pattern: length = 10

550: AAGTAGTGTG 9635: AAGTAGTGTG 8816: AAGTAGTGTG

===70th pattern: length = 10

188: GAGAACACCA 9273: GAGAACACCA 8974: CACAAGAGGA

===71st pattern: length = 10

8422: AGAAGAAGAA 8979: GAGGAGGAGG 8982: GAGGAGGAGG

===72nd pattern: length = 11

611: GTCAGTGTGGA 9696: GTCAGTGTGGA 9001: CAGTCACACCT

===73rd pattern: length = 10

612: TCAGTGTGGA 9697: TCAGTGTGGA 9002: AGTCACACCT

【 0 1 4 3 】

第 1 実施例で発見された 7 3 個のパターンのうちの 2 2 個のパターンは通常の接尾辞木で発見できる通常の繰り返しのみのパターンであるが、残りの 5 1 個のパターンは本発明を適用せずに発見することは困難である。

【 0 1 4 4 】

〔第 2 実施例〕

また本願発明者は、第 1 実施例と同じ配列に対し、本発明を適用して長さ 8 以上でかつ出現回数が 7 回以上の全てのパターンを検索する実験を行った。検索結果を以下に示す。

=== 1st pattern: length = 8

54: ACCACACA 9139: ACCACACA 780: AGGAGAGA 184: AGGAGAGA
9269: AGGAGAGA 6121: GTTGTGTG 569: GTTGTGTG 9654: GTTGTGTG
2166: GAAGAGAG 6676: AGGAGAGA

=== 2nd pattern: length = 8

58: CACACAAG 9143: CACACAAG 1472: GAGAGAAC 6352: TGTGTGGA
186: GAGAGAAC 9271: GAGAGAAC 5324: CACACAAG

=== 3rd pattern: length = 8

91: CACACCAG 9176: CACACCAG 160: AGAGAAGT 6124: GTGTGGTC
7153: GAGAGGAC 242: AGAGAAGT 9327: AGAGAAGT 3193: CACACCAG
7385: ACACAACT

=== 4th pattern: length = 8

71: TTCCCTGA 9156: TTCCCTGA 261: GGTTTGAC 9346: GGTTTGAC
347: AAGGGACT 9432: AAGGGACT 5280: TTGGGTCA

=== 5th pattern: length = 8

99: GGCCAGGG 2119: GGCCAGGG 383: GGCCTGGG 9468: GGCCTGGG
853: GGCCAGGG 4885: AATTCAAA 9184: GGCCAGGG

=== 6th pattern: length = 8

296: CGAGAGCT 9381: CGAGAGCT 588: TAGAGATC 9673: TAGAGATC
6964: TACACATG 624: ATCTCTAG 9709: ATCTCTAG

=== 7th pattern: length = 8

508: ACCCACTG 9593: ACCCACTG 594: TCCCTCAG 9679: TCCCTCAG
3699: GAAAGACT 5287: AGGGAGTC 2250: TCCCTCAG

=== 8th pattern: length = 8

411: CCCTCAGA 9496: CCCTCAGA 595: CCCTCAGA 9680: CCCTCAGA
2204: CCCTCAGA 3700: AAAGACTC 5288: GGGAGTCT

=== 9th pattern: length = 8

56: CACACACA 9141: CACACACA 239: GAGAGAGA 9324: GAGAGAGA
782: GAGAGAGA 8436: GAGAGAGA 8437: AGAGAGAG 8438: GAGAGAGA

===10th pattern: length = 8

227: ATGGATGA 9312: ATGGATGA 700: GCTTGCTG 1532: ATGGATGA
897: GCAAGCAG 1136: GCAAGCAG 3098: ATGGATGA

===11th pattern: length = 8

630: AGCAGTGG 946: CATCAGAA 1535: GATGACAA 7887: TGCTGAGG
3209: CATCAGAA 7756: AGCAGTGG 6004: CATCAGAA

===12th pattern: length = 8

398: GGGGAGTG 9483: GGGGAGTG 1070: AAAAGACA 3462: AAAACAGA
6591: CCCCACTC 2743: AAAAGACA 3292: AAAAGACA 5044: AAAACAGA
7249: AAAACAGA

===13th pattern: length = 8

786: GAGATGGG 1106: AGAGCAAA 6256: GAGATGGG 6268: GAGATGGG
7226: AGAGCAAA 2832: CACATCCC 5205: CACATCCC

===14th pattern: length = 8

863: AAAGAAAA 2851: AAAGAAAA 2737: AAAGAAAA 6167: AAAGAAAA
1112: AAACAAAA 3065: AAACAAAA 4526: TTTCTTTT 4702: AAAGAAAA
4785: AAAGAAAA 9069: AAAGAAAA

===15th pattern: length = 8

865: AGAAAAAA 2632: AGAAAAAA 2853: AGAAAAAA 1123: AGAAAAAA
2739: AGAAAAAA 3201: ACAAAAAA 7747: AGAAAAAA 1359: GTGGGGGG

===16th pattern: length = 8

1258: AAGTAGTA 8868: CCAGCAGC 1502: AACTACTA 6047: AAGTAGTA
6296: TTGATGAT 8715: AAGTAGTA 6471: AAGTAGTA 7683: AAGTAGTA

===17th pattern: length = 8

294: CCCGAGAG 9379: CCCGAGAG 622: AAATCTCT 9707: AAATCTCT
1665: TTTAGAGA 8072: AAATCTCT 1983: GGGCACAC

===18th pattern: length = 8

756: AAATTTTG 1787: TTAAAAAG 2005: GGGCCCCT 4891: AAATTTTC
2542: AAATTTTC 4781: TTAAAAAG 9065: TTAAAAAG 7246: TTAAAAAC

===19th pattern: length = 8

259: GAGGTTTG 9344: GAGGTTTG 345: ACAAGGGA 9430: ACAAGGGA
2035: GTGGAAAG 2110: ACAAGGGA 3841: AGAACCCA 7970: GTGGAAAG

===20th pattern: length = 8

566: TCTGTTGT 9651: TCTGTTGT 2191: AGACAACA 3434: ACAGAAGA
2627: ACAGAAGA 3164: AGACAACA 7734: GAGTGGTG

===21st pattern: length = 8

169: AGAAGAAG 3436: AGAAGAAG 2193: ACAACAAC 7021: AGAAGAAG
7635: GAGGAGGA 8422: AGAAGAAG 8979: GAGGAGGA 8982: GAGGAGGA
8425: AGAAGAAG

===22nd pattern: length = 8

1358: AGTGGGGG 3200: GACAAAAA 2199: ACTCCCCC 3288: CAGAAAAA
3040: GACAAAAA 6315: CAGAAAAA 3936: GACAAAAA 5378: CTGTTTTT

===23rd pattern: length = 8

1915: TAATGATG 2476: ATTAGTAG 2647: ATTAGTAG 2767: ATTAGTAG
3634: TAATGATG 7220: ATTAGTAG 6093: ATTAGTAG

===24th pattern: length = 8

1124: GAAAAAAG 4792: AGGGGGGA 4801: TGGGGGGT 2740: GAAAAAAG
3202: CAAAAAAC 7748: GAAAAAAG 9076: AGGGGGGA

===25th pattern: length = 8

2036: TGGAAAGG 2111: CAAGGGAA 3005: TGGAAAGG 8775: TGGAAAGG
4931: TGGAAAGG 2900: GTTCCCTT 4955: TGGAAAGG 7781: CTTGGGTT

===26th pattern: length = 8

864: AAGAAAAA 2631: AAGAAAAA 2852: AAGAAAAA 1122: AAGAAAAA
2738: AAGAAAAA 3145: AACAAAAA 6168: AAGAAAAA 7132: AAGAAAAA

===27th pattern: length = 8

2128: ATTTTCTT 2157: GCCCCACC 3477: TAAAAGAA 3642: TAAAACAA
7291: TAAAACAA 4524: ATTTTCTT 4783: TAAAAGAA 9067: TAAAAGAA
7482: TAAAACAA 7529: GCCCCTCC

===28th pattern: length = 8

373: AGGGAGGC 9458: AGGGAGGC 4820: GAAAGAAT 6733: GAAAGAAT
3793: GTTTGTTA 8797: TGGGTGGC 8847: GAAAGAAT

===29th pattern: length = 8

1111: AAAACAAA 2850: AAAAGAAA 3064: AAAACAAA 4525: TTTTCTTT
4784: AAAAGAAA 9068: AAAAGAAA 3804: CCCCTCCC 4327: AAAAGAAA
4870: AAAACAAA 7483: AAAACAAA 7530: CCCCTCCC 8587: GGGGTGGG

===30th pattern: length = 8

587: CTAGAGAT 9672: CTAGAGAT 6963: GTACACAT 4422: GTACACAT
6576: CATGTGTA 5202: GTACACAT 5467: GATCTCTA

===31st pattern: length = 8

282: TTCATCAC 550: AAGTAGTG 9635: AAGTAGTG 8816: AAGTAGTG
4496: CCAGCAGA 5001: AAGTAGTG 9367: TTCATCAC

===32nd pattern: length = 8

755: AAAATTTT 1786: TTTTAAAA 4890: AAAATTTT 4780: TTTTAAAA
9064: TTTTAAAA 4530: TTTTAAAA 6495: AAAATTTT

===33rd pattern: length = 8

374: GGGAGGCC 9459: GGGAGGCC 4560: AAACAATA 4821: AAAGAATA
8644: AAAGAATA 6734: AAAGAATA 7293: AAACAATA

===34th pattern: length = 8

1324: CCACCCCA 8181: AAGAAAAG 4786: AAGAAAAG 9070: AAGAAAAG
7358: GGAGGGGA 5014: AAGAAAAG 7319: GGAGGGGA

===35th pattern: length = 8

2788: TAATAAGA 5497: TAATAACA 6870: ATTATTGT 5233: TAATAACA
5736: TAATAAGA 6910: TAATAAGA 7852: ATTATTGT

===36th pattern: length = 8

6368: ACCACCAC 7022: GAAGAAGA 7636: AGGAGGAG 7120: CAACAACA
8423: GAAGAAGA 8980: AGGAGGAG 8983: AGGAGGAG

===37th pattern: length = 8

84: AGAACTAC 9169: AGAACTAC 4186: ACAAGTAG 4384: ACAAGTAG
5327: ACAAGTAG 7556: TGTTCATC 8925: ACAAGTAG

【 0 1 4 5 】

〔第 3 実施例〕

Streptococcus anginosus という細菌の 1 6 S 部分の配列（アクセス番号：SA1
6SRRNA，長さ：1334）を以下に示す。

gaacgggtgagtaacgcgtaggtaacctgcctattagaggggataactatttgaaacga

tagctaataccgcataacagtatgtaacacatgttagatgcttgaaagatgcaattgcat
cgctagtagatggacctgcgttgatttagctagtaggtagggtaaaggcctacctaggca
acgatacatagccgacctgagaggggtgatcgggccacactgggactgagacacggcccaga
ctcctacgggaggcagcagtagggaaatcttcggcaatggggggaaccctgaccgagcaac
gccgcgtgagtgaagaaggttttcggatcgtaaagctctgttgtaaggaagaacgagt
tgagaatggaaagttcatactgtgacggtacttaaccagaaagggacggctnactacgtg
ccagcagccgcggttaatacgttaggtcccnagcgttggtccgatttattgggcgtaaagcg
agcgcaggcgggttagaaaagtcgaagtgaaggcagtggtcaaccattgtaggctttg
gaaactgtttaacttgagtgcagaaggggagagtggaaattccatgtgtagcggtgaaatg
cgtagatataatggaggaacaccggtggcgaaagcggctctctggtctgtaactgacgctg
aggctcgaaagcgtggggagcgaacaggattagataccctngtagtccacgccgtaaacg
atgagtgcctaggtgttggttcctttccgggactcagtgccgcagctaaccgattaagcac
tccgcctggggagtacgaccgcaaggttgaaactcaaaggaattgacgggggccgcacaa
gcggtggagcatgtngtttaattcgaagnaacgcgaagaaccttaccaggtcttgacatc
ccgatgctntttctagagataggaagtttcttcggaacatcggtgacaggtggtgcatgg
ttgtcgtcagctcgtgtcgtgagatgttggttaagtcccgcaacgagcgcaacctnat
tgttagttgccatcattaagttgggcactctagcgagactgccggtaatnaaccggagga
aggtggggatgacgtcaaatcatcatgcccccttatgacctnggctacacacgtgctacaa
tggtctgttacaacgagtcgcaagccggtgacggcaagctaattctctgaaagccagtcctca
gttcggattgtaggctgcaactcgccctncatgaagtcggaatcgctagtaatcgcggatc
agcacgccgcggtgaatacgttcccgggccttgtagacaccgcncgtcacaccacgagag
tttgtaacacccga

【0 1 4 6】

本願発明者は、上記の配列に対し、本発明を適用して長さ 10 以上でかつ出現回数が 2 回以上の全てのパターンを検索する実験を行った。検索結果を以下に示す。なお、第 3 実施例では検索対象の配列が第 1 実施例及び第 2 実施例で用いた配列より短いため、検索されたパターンの数は少なくなっている。

=== 1st pattern: length = 10

131: GGACCTGCGT 370: AAGTTCATAC


```

=== 2nd pattern: length = 10
    97: ATGCTTGAAA    438: CGTAGGTCCC
=== 3rd pattern: length = 11
    280: GGGAAACCCTGA  536: TTTGGAAACTG
=== 4th pattern: length = 10
    29: CCTATTAGAG   630: AAGCGGCTCT
=== 5th pattern: length = 10
    739: TCCTTTCCGG   985: GTTGGGTAA
=== 6th pattern: length = 10
    527: ATTGTAGGCT  1206: ATTGTAGGCT
=== 7th pattern: length = 10
    118: ATCGCTAGTA  1240: ATCGCTAGTA

```

【 0 1 4 7 】

【発明の効果】

以上説明したように本発明は、配列中に存在している他の構成要素へ置換可能な変数を、前記変数と相補の関係を有する別の変数の位置を指し示す情報又は前記別の変数が存在していないことを表す情報に変換することを、処理対象の配列中に存在している全ての変数について行って、処理対象の配列を変換するので、配列の構造を効率良く解析できるように配列を変換することができる、という優れた効果を有する。

【 0 1 4 8 】

また本発明は、配列中に存在している他の構成要素へ置換可能な変数を、同一の変数の位置を指し示す情報又は同一の変数が存在していないことを表す情報に変換することで処理対象の配列を第 1 の配列へ変換すると共に、配列中に存在している他の構成要素へ置換可能な変数を、前記変数と相補の関係を有する別の変数の位置を指し示す情報又は前記別の変数が存在していないことを表す情報に変換することで処理対象の配列を第 2 の配列へ変換し、第 1 の配列及び第 2 の配列を用いて処理対象の配列の構造を解析するので、配列の構造解析を効率良く行うことができる、という優れた効果を有する。

【図面の簡単な説明】

【図 1】 本実施形態に係るコンピュータシステムの概略構成を示すブロック図である。

【図 2】 本実施形態に係る構造接尾辞木生成処理の内容を示すフローチャートである。

【図 3】 prev()演算処理の内容を示すフローチャートである。

【図 4】 compl()演算処理の内容を示すフローチャートである。

【図 5】 構造接尾辞木作成処理の内容を示すフローチャートである。

【図 6】 (A)乃至(H)は本実施形態に係る構造接尾辞木の作成過程の一例を示す概念図である。

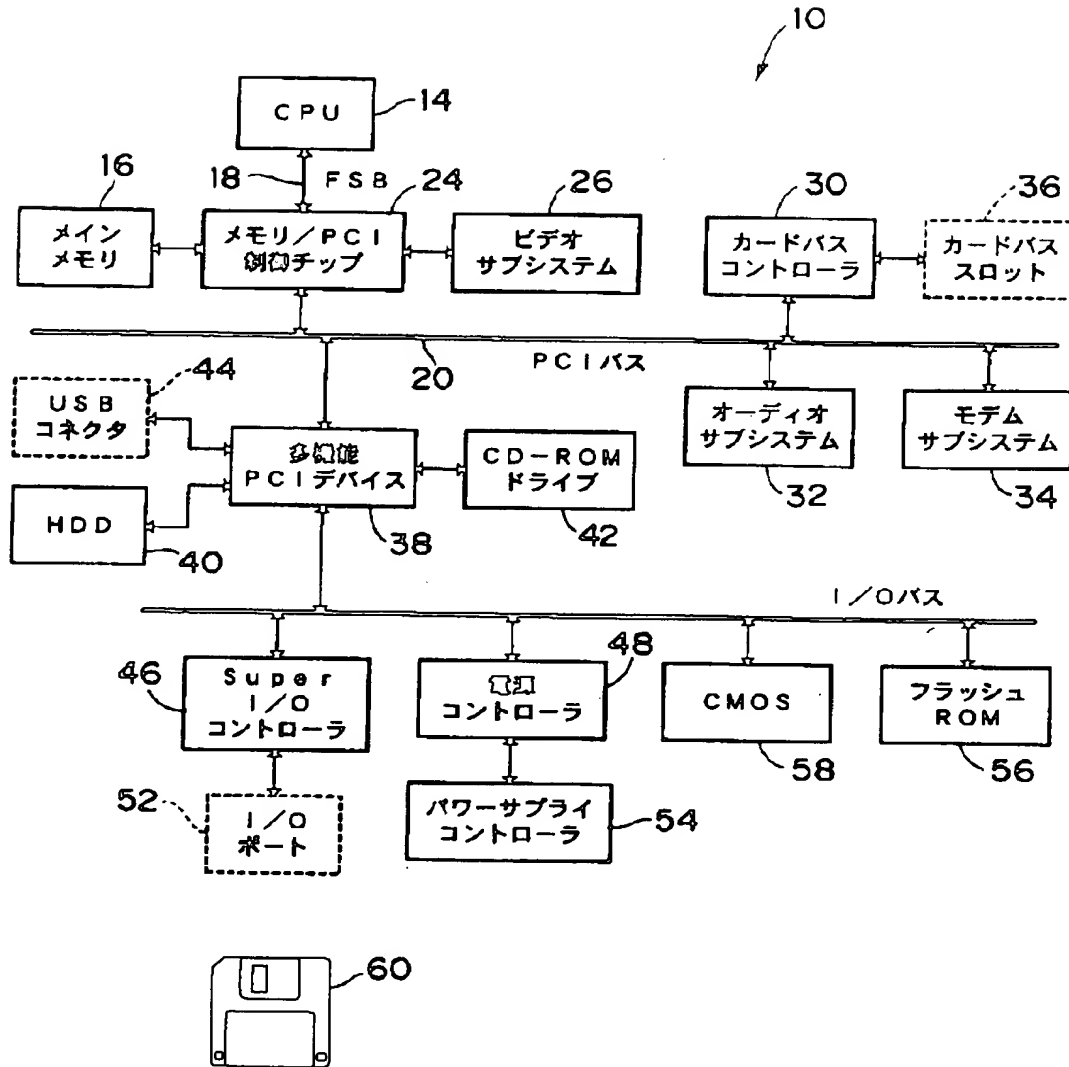
【図 7】 接尾辞木の一例を示す概念図である。

【符号の説明】

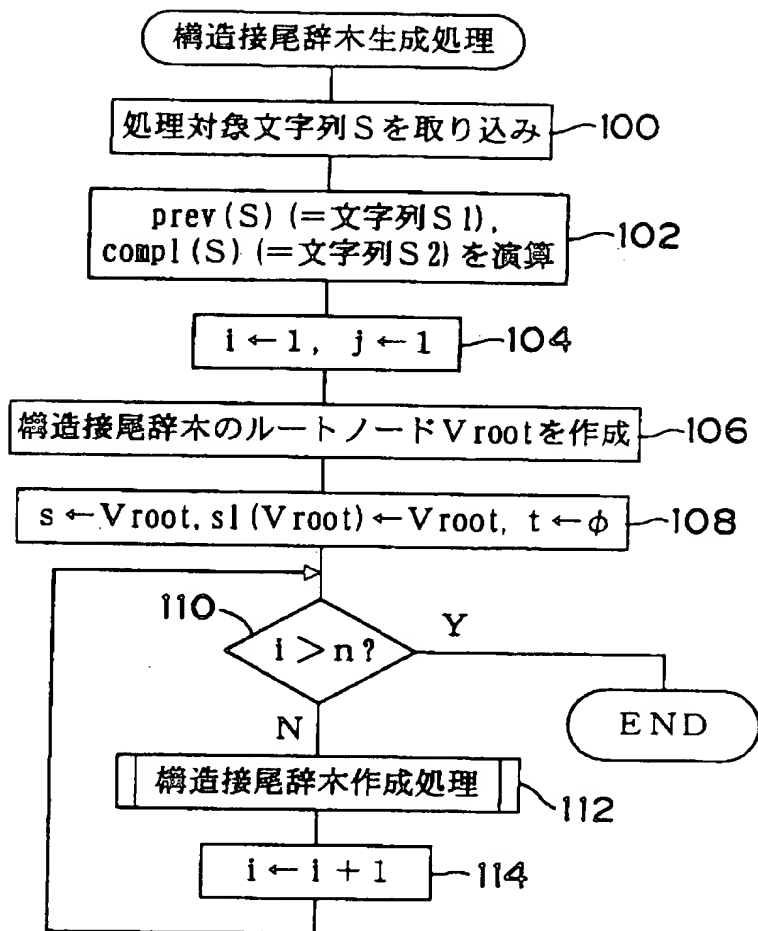
1 0	コンピュータシステム
1 4	C P U
6 0	情報記憶媒体

【書類名】 図面

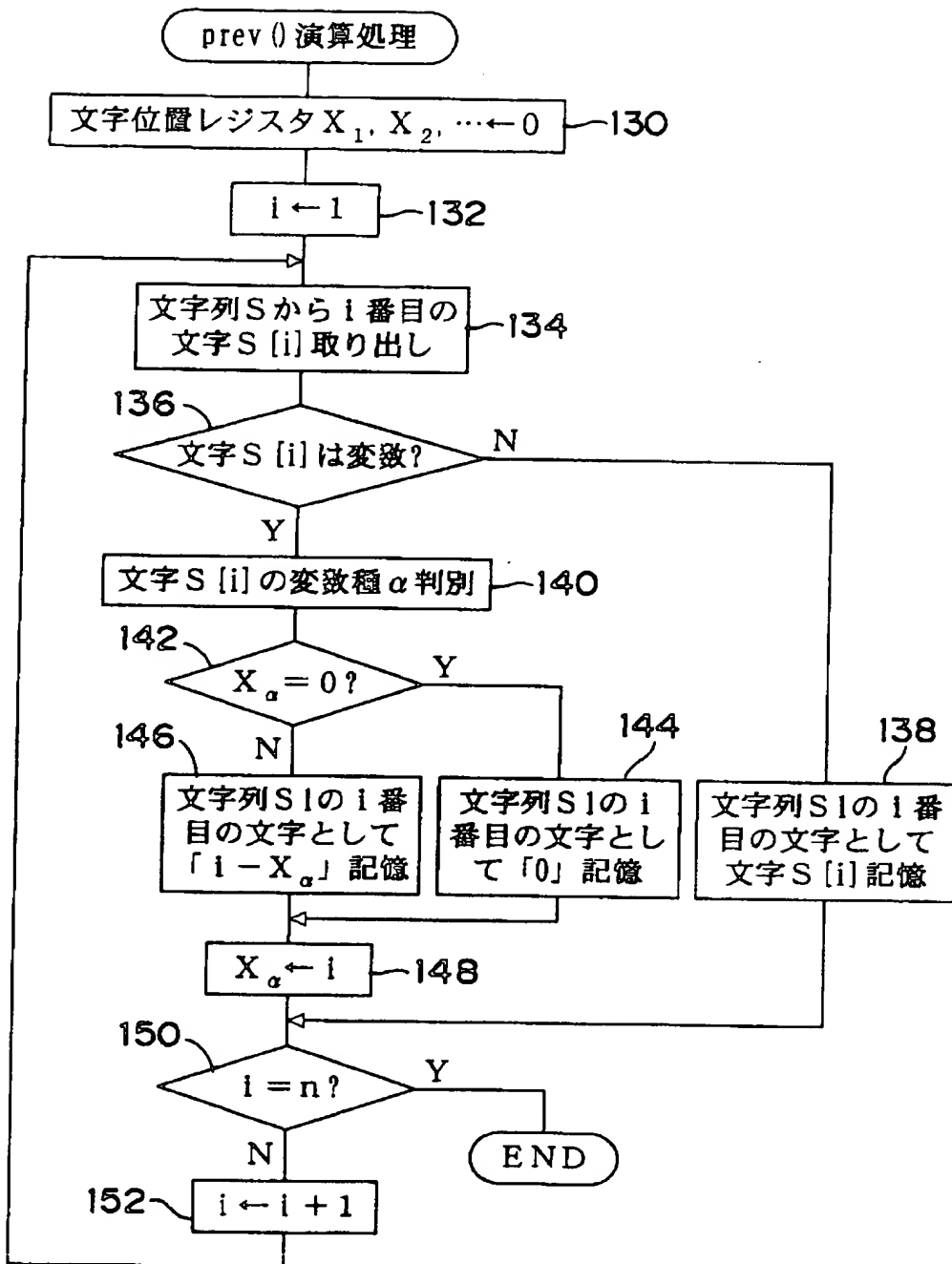
【図 1】



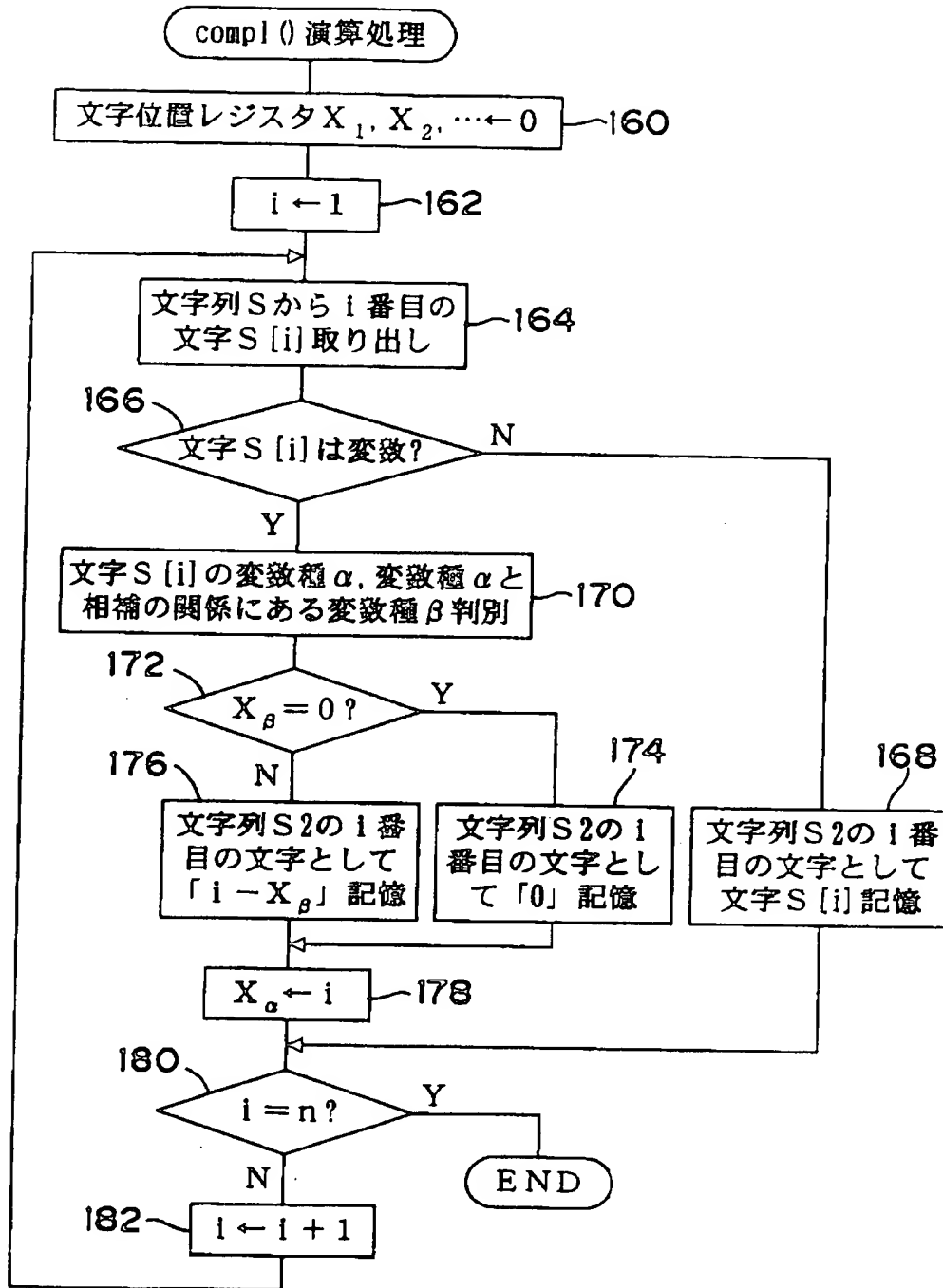
【図 2】



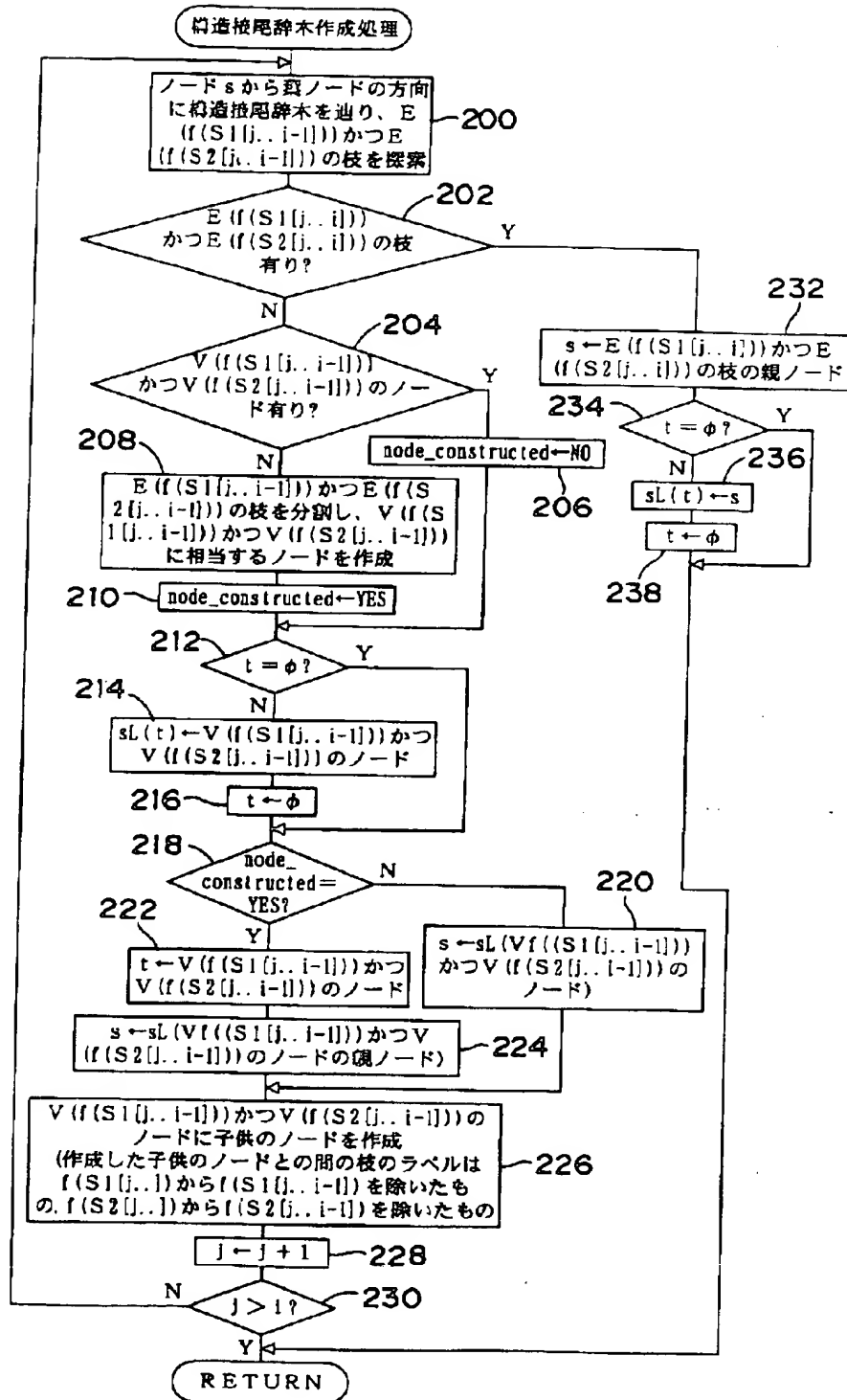
【図 3】



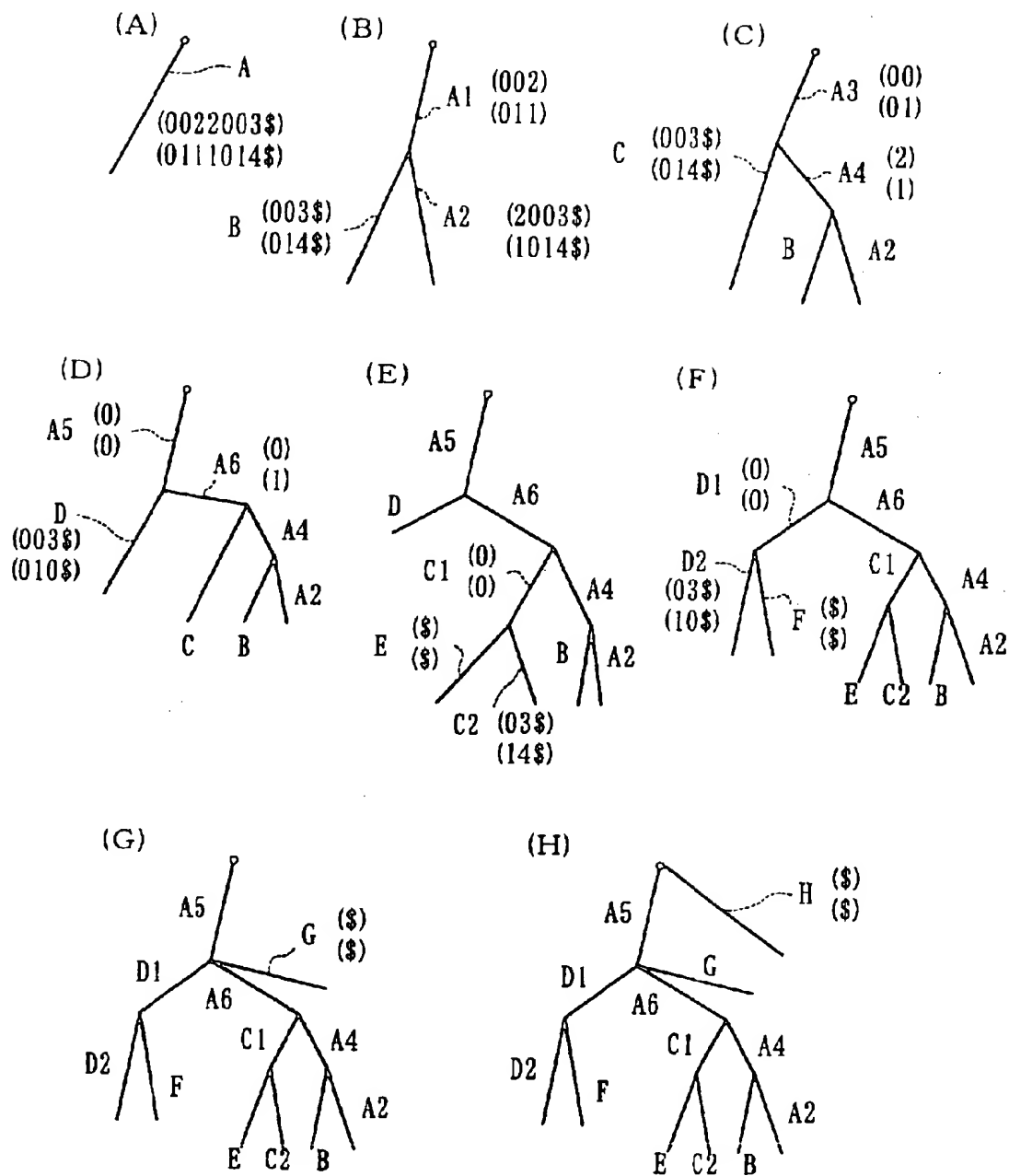
【図 4】



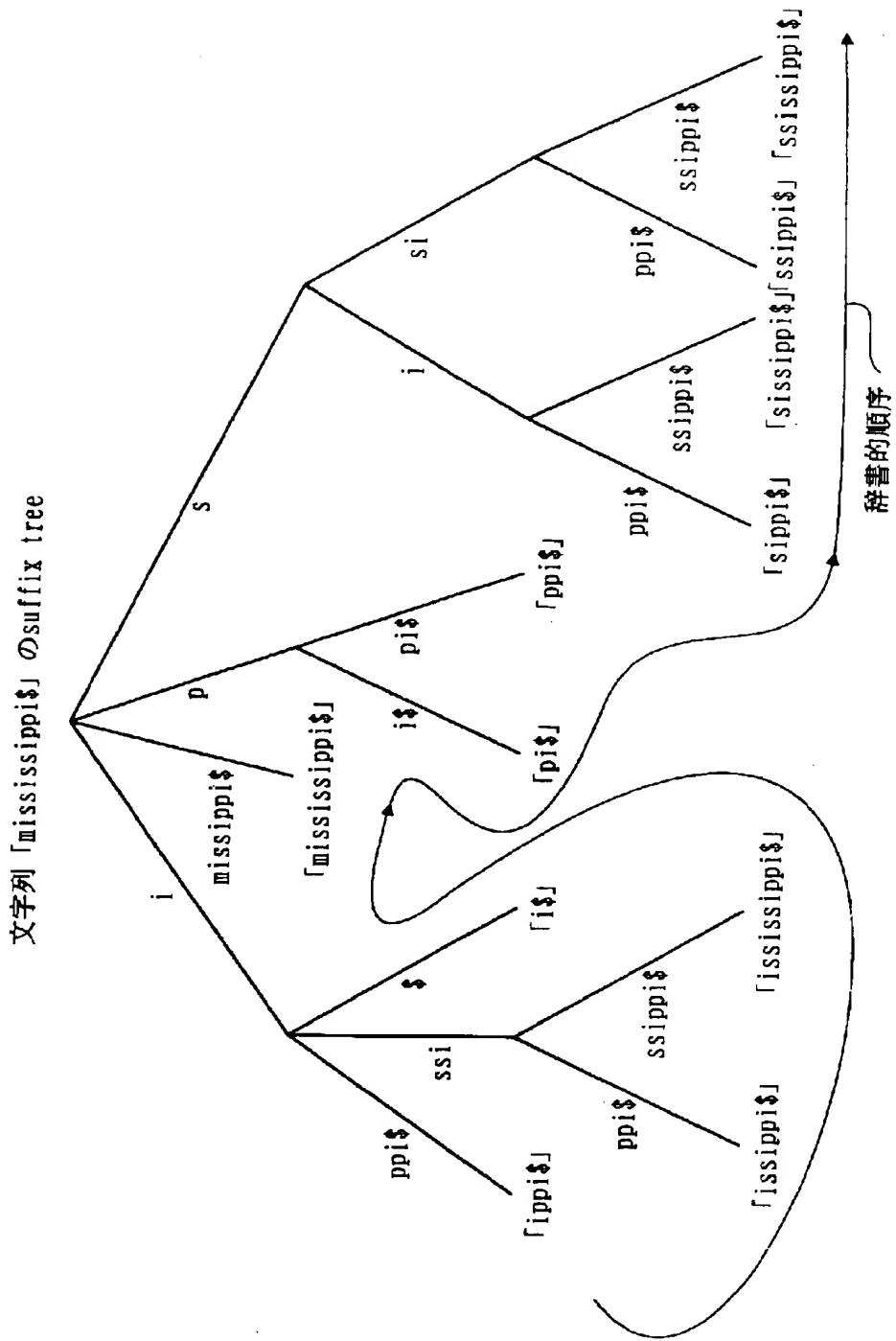
【図 5】



【図 6】



【图 7】



【書類名】 要約書

【要約】

【課題】 配列の構造解析を効率良く行う。

【解決手段】 処理対象の文字列 S に対して $\text{prev}(S)$ を演算することで、文字列 S 中の全ての変数を、変数よりも上流側に同一の変数が存在していれば該同一の変数との隔たりを表す数値に変換し、変数よりも上流側に同一の変数が存在していなければ「0」に変換した文字列 $S1$ を求めると共に、 $\text{compl}(S)$ を演算することで、文字列 S 中の全ての変数を、変数よりも上流側に相補の関係を有する変数が存在していれば該相補の関係を有する変数との隔たりを表す数値に変換し、変数よりも上流側に該相補の関係を有する変数が存在していなければ「0」に変換した文字列 $S2$ を求める(102)。そして文字列 $S1, S2$ を対応する一組の文字列とみなして単一の接尾辞木（構造接尾辞木）を生成し（104～114）、構造接尾辞木を用いて文字列 S が表す配列の構造を解析する。

【選択図】 図 2

認定・付加情報

特許出願の番号	平成 1 1 年 特許願 第 3 6 8 4 2 0 号
受付番号	5 9 9 0 1 2 6 5 9 6 1
書類名	特許願
担当官	塩崎 博子 1 6 0 6
作成日	平成 1 2 年 2 月 9 日

<認定情報・付加情報>

【特許出願人】

【識別番号】	390009531
【住所又は居所】	アメリカ合衆国 1 0 5 0 4、ニューヨーク州 アーモンク (番地なし)
【氏名又は名称】	インターナショナル・ビジネス・マシーンズ・コーポレーション

【代理人】

【識別番号】	100086243
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	坂口 博

【代理人】

【識別番号】	100091568
【住所又は居所】	神奈川県大和市下鶴間 1 6 2 3 番地 1 4 日本アイ・ビー・エム株式会社 大和事業所内
【氏名又は名称】	市位 嘉宏

【復代理人】

【識別番号】	申請人
【識別番号】	100079049
【住所又は居所】	東京都新宿区新宿 4 丁目 3 番 1 7 号 HK新宿ビル 7 階 太陽国際特許事務所
【氏名又は名称】	中島 淳

【選任した復代理人】

【識別番号】	100084995
【住所又は居所】	東京都新宿区新宿 4 丁目 3 番 1 7 号 HK新宿ビル 7 階 太陽国際特許事務所
【氏名又は名称】	加藤 和詳

【選任した復代理人】

【識別番号】	100085279
--------	-----------

次頁有

認定・付加情報（続き）

【住所又は居所】	東京都新宿区新宿四丁目 3 番 1 7 号	HK 新宿ビル 7 階	太陽国際特許事務所
【氏名又は名称】	西元	勝一	
【選任した復代理人】			
【識別番号】	100099025		
【住所又は居所】	東京都新宿区新宿 4 丁目 3 番 1 7 号	HK 新宿ビル 7 階	太陽国際特許事務所
【氏名又は名称】	福田	浩志	

出 願 人 履 歴 情 報

識別番号 [390009531]

1. 変更年月日 1990年10月24日
[変更理由] 新規登録
住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション
2. 変更年月日 2000年 5月16日
[変更理由] 名称変更
住 所 アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
氏 名 インターナショナル・ビジネス・マシーンズ・コーポレーション